AD-A221 429

4

DTIC FILE COPY

AD______________

REPORT NO T2-90

COMPUTERIZED ANALYSIS OF NUTRIENTS (CAN) SYSTEM

U S ARMY RESEARCH INSTITUTE
OF
ENVIRONMENTAL MEDICINE
Natick, Massachusetts

NOVEMBER 1989

UNITED STATES ARMY
MEDICAL RESEARCH & DEVELOPMENT COMMAND

90 05 09 154

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION N/A	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release Distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION U.S.A. Research Institute of Environmental Medicine 6b. OFFICE SYMBOL (If applicable) SGRD-UE-NR	7a. NAME OF MONITORING ORGANIZATION U.S.A. Medical Research and Development Command
6c. ADDRESS (City, State, and ZIP Code) Natick, MA 01760-5007	7b. ADDRESS (City, State, and ZIP Code) Fort Detrick Frederick, MD 21701-5012
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
63002D	3M2630 02D819	AI	DA305222

11. TITLE (Include Security Classification)

COMPUTERIZED ANALYSIS OF NUTRIENTS (CAN) SYSTEM

12. PERSONAL AUTHOR(S) M.S. Rose, J. Finn, C. Radovsky, M. Benson, K. Samonds, D. Poe, M. Sutherland, W. Wisnaskas, C. Baker, D. Sherman, E. W. Askew

13a. TYPE OF REPORT	13b. TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Oct 88 TO Oct 89	1989 November 2	354

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Nutritional Analysis System, Computer, User's Guide, Diet History Analysis, Recipe Analysis, CAN System. [illegible]

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This technical report constitutes a manual that describes the process the Military Nutrition Division (MND) of the U.S. Army Research Institute of Environmental Medicine uses to collect and analyze dietary intake data gathered in garrison and field nutrition research studies. The Computerized Analysis of Nutrients (CAN) system includes a nutrient database, recipe analysis component, diet history analysis program, food consumption and other biochemical data entry software, and programs for combining food consumption and database information. The University of Massachusetts Nutrient Data Bank was used to analyze dietary intake in previous MND research studies (1985-1989). The USDA Standard Reference tapes and the Continuing Survey of Food Intakes by Individuals (CSFII) are the basis of the new CAN system database. Unlike other food analysis systems, the CAN system allows for the use of USDA retention factors from CSFII in order to estimate nutrient losses resulting from different cooking methods. This technical report will be updated as major revisions to the different programs occur and will serve as documentation for MND nutrient analysis procedures.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Madeleine S. Rose, LTC, AMSC	22b TELEPHONE (Include Area Code) (508) 651-4979 22c. OFFICE SYMBOL SGRD-UE-NR

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

COMPUTERIZED ANALYSIS OF NUTRIENTS (CAN) SYSTEM

LTC Madeleine S. Rose[1]
John Finn[2]
Carlo Radovsky[1]
Michael Benson[2]
Dr. Kenneth Samonds[3]
Dr. Donald Poe[4]
Dr. Michael Sutherland[4]
William Wisnaskas[5]
Carol Baker[1]
Doris Sherman[1]
LTC Eldon W. Askew[1]

[1]U.S. Army Research Institute of Environmental Medicine
Natick, MA 01760

[2]GEO-CENTERS, INC.
Newton, MA 02159

[3]University of Massachusetts
Amherst, MA 01002

[4]Statistical Computing Center
University of Massachusetts
Amherst, MA 01002

[5]Northeastern University
Boston, MA 02117

November 1989

TABLE OF CONTENTS

FOREWORD ix

PREFACE xi

CHAPTER 1.0 NUTRITIONAL ANALYSIS SYSTEM FLOW CHART 1
- 1.1 Introduction 3
- 1.2 Flowcharts 3

CHAPTER 2.0 NUTRIENT DATABASE 17
- 2.1 Nutrient Database Flow Charts 19
- 2.2 Checklist of Features of Nutrient Database 19
- 2.3 Transfer of Military Ration Data From Food Engineering Directorate (FED) 20
- 2.4 ORACLE Relational Database System 20

CHAPTER 3.0 RECIPE ANALYSIS 25
- 3.1 Nutrition Project Report 27
  - 3.1.1 Introduction 27
  - 3.1.2 Software Development 27
  - 3.1.3 Documentation 28
- 3.2 Recipe Coder / Editor / Analyzer User's Guide 28
  - 3.2.1 Introduction 28
  - 3.2.2 Getting Started 28
  - 3.2.3 The Main Menu 30
    - 3.2.3.1 Creating and Appending to Recipe Files 31
    - 3.2.3.2 Checking a Coded Recipe File 36
    - 3.2.3.3 Editing a Recipe 37
      - 3.2.3.3.1 Line by Line Recipe Editing 39
    - 3.2.3.4 Creating Recipe Sub-set Files 42
    - 3.2.3.5 Combining Recipe Files 43
    - 3.2.3.6 Analyzing Recipe Files 44
- 3.3 Recipe Coder / Editor / Analyzer Programmer's Guide 48
  - 3.3.1 Introduction 48
  - 3.3.2 File Formats 48
  - 3.3.3 Overall Program Structure 54
  - 3.3.4 Recipe Analysis 56
- 3.4 Oracle Table Translator (Dumptable) 61
  - 3.4.1 Introduction 61
  - 3.4.2 Running the Program 62
  - 3.4.3 Oracle Table Formats 62
  - 3.4.4 Programming Considerations 64
- 3.5 Menu Manager Programmer's Guide 66
  - 3.5.1 Introduction 66
  - 3.5.2 Resource Files and Screen Forms 67
  - 3.5.3 Selection Menus: 71

3.5.5 Using Menus in a Program 77
3.5.6 More about Using Menus in a Program 82
3.5.7 Related Topics 86
3.6 Recipe Coder / Editor / Analyzer Program Details 95
3.6.1 Introduction 95
3.6.2 Program Files 95
3.6.3 Subroutines & Functions 97
3.7 Using the Oracle Data Loader for the Recipe Coder / Editor / Analyzer 100
3.7.1 Introduction 100
3.7.2 Creating an Oracle Table 100
3.7.3 Dumping Analyzed Output into the Oracle Table 102

CHAPTER 4.0 NUTRITION DATA ENTRY PROGRAMS 107
4.1 Food Frequency Questionnaire 109
4.2 Food Consumption Data Entry User's Guide 109
4.2.1 Introduction 109
4.2.2 Getting Started 110
4.2.3 The Main Menu 110
4.2.3.1 Enter Food Data 111
4.2.3.2 Edit Food Data 113
4.2.3.3 Produce a Report 115
4.2.3.4 Create/Print Template 115
4.2.3.5 Default Values 120
4.3 Food Consumption Data Entry Programmer's Guide 121
4.3.1 Introduction 121
4.3.2 Program Source Code Files 122
4.3.3 Subroutines and Functions 122
4.3.4 Program Data Structures 124
4.3.5 Program Operation 124
4.4 Diet History Analyzer User's Guide 125
4.4.1 Introduction 125
4.4.2 Getting Started 125
4.4.3 The Main Menu 126
4.4.3.1 Creating a Diet History File 127
4.4.3.2 Appending to an Existing Diet History File 129
4.4.3.3 Checking a Coded Diet History File 129
4.4.3.4 Editing a Diet History 129
4.4.3.4.1 Line by Line Diet History Editing 130
4.4.3.5 Combining Diet History Files 132
4.4.3.6 Analyzing Diet Histories 132
4.5 Diet History Analyzer Programmer's Guide 135
4.5.1 Introduction 135
4.5.2 Program Files 135
4.5.3 Subroutines & Functions 136
4.5.4 Program Operation 138
4.5.5 Diet History Analysis 139

4.5.6 File Formats 140
4.6 Universal Data Entry Program 141
4.6.1 Enter Data Into A New File 142
4.6.2 Enter Data Into An Existing File 146
4.6.3 Edit An Existing Data File 146
4.6.4 Produce a Report From An Existing File 148
4.6.5 Add New Variables to a File 150
4.6.6 Rearrange The Variables in The Data File 153
4.6.7 Change the Size of Numeric Variables 154

CHAPTER 5.0 FOOD CONSUMPTION ANALYSIS 157
5.1 Guideline for Consumption Data Analysis 159
5.1.1 Setup and Organization of the VAX 159
5.1.1.1 General Information 159
5.1.1.2 Command Usage on the VAX 161
5.1.1.3 Creating a Directory 162
5.1.1.4 Methods of Moving Around the VAX 163
5.1.2 Useful Features and Programs on the VAX 163
5.1.2.1 VAX Features 163
5.1.2.1.1 Use of the Help Utility 166
5.1.2.1.2 Directory 167
5.1.2.1.3 Search 172
5.1.2.1.4 Differences 174
5.1.2.1.5 Set Protection 175
5.1.2.1.6 Sort 179
5.1.2.1.7 Rename 181
5.1.2.1.8 Copy 183
5.1.2.1.9 Append 185
5.1.2.2 VAX Programs 186
5.1.2.2.1 The See Program 187
5.1.2.2.2 The Rearrange Program 188
5.1.2.2.3 The Fixform Program 189
5.1.3 File Transfer from the PC to the VAX 190
5.1.4 Cleanup, Merging, and Analysis of Data 191
5.1.4.1 General Information 191
5.1.4.2 General Setup of Data 192
5.1.4.2.1 COD File Setup 192
5.1.4.3 Useful Programs and Their Results 193
5.1.4.3.1 CHANGE Program 193
5.1.4.3.2 ANALYZE Program 194
5.1.4.3.3 The Nutrient Table File 196
5.1.4.3.4 Statistical Analysis 197
5.1.5 Standard Statistical Procedures 197
5.1.5.1 General Information 197
5.1.5.2 Information on SPSSx 198
5.1.5.2.1 Running_SPSSx 198
5.1.5.2.2 Default_File_Extensions 199

5.1.5.3 Stages of Analysis 199
5.1.6 Documentation of a Study's Directories 201
5.1.6.1 General Information 201
5.1.6.2 Docfiles Program 201
5.2 Analyze Consumption Data User's Guide 201
5.2.1 Introduction 201
5.2.2 Getting Started 202
5.2.3 The Main Menu 203
5.2.3.1 Change Consumption Data 203
5.2.3.2 Analyze the Consumption Data 204
5.3 EXAMPLE: Alaska Consumption Data Analysis Procedures 205
5.3.1 Dealing with Corrected Raw Data 205
5.3.2 Description of the CHANGE Program 206
5.3.3 Analyze the Consumption Data 207
5.3.4 Running Statistical Analysis on the Data 208
5.3.5 Description of Alaska Data Files 209
5.3.5.1 The COD File 209
5.3.5.2 The Field Data Files 209
5.3.5.3 The NAMES.LIS File 210
5.3.5.4 The Merged Data File: FOODS.TOT 210
5.3.5.5 The RATING.DAT File 211

CHAPTER 6.0 JOB CONTROL LANGUAGE SAMPLES
FOR STATISTICAL ANALYSES 213

CHAPTER 7.0 MASTER FILE OF A RATION RECIPES 263

CHAPTER 8.0 COMPARISON OF CAN, UMASS and LSU
NUTRITIONAL ANALYSIS SYSTEMS 299

APPENDIX A 305

APPENDIX B 307

APPENDIX C 311

APPENDIX D 313

APPENDIX E 315

APPENDIX F 317

APPENDIX G 319
G-1 General Information 319
G-1.1 Variables 319
G-1.2 Formats 320
G-1.3 Variable Identifiers 320

G-1.4 Input/Output 321
G-1.5 Other Considerations 322
G-2 Technical Aspects of the COD File 322
G-3 The Change Program 324
G-3.1 General Information 324
G-3.2 Variable Declaration Section 324
G-3.3 Data Initialization Section 324
G-3.4 Data Processing Section 325
G-3.5 Format Specification Section 326

FOREWORD

This technical manual provides information on the process that the Military Nutrition Division (MND) of the U.S. Army Research Institute of Environmental Medicine (USARIEM) uses to collect and analyze dietary intake data gathered in garrison and field nutrition research studies. The Computerized Analysis of Nutrients (CAN) system includes: a nutrient database, recipe analysis component, diet history analysis program, food consumption and other biochemical data entry software, and programs for combining food consumption and database information. This technical report is a joint effort of the Military Nutrition Division at USARIEM, GEO-CENTERS, INC., and the University of Massachusetts, Amherst, MA.

The University of Massachusetts Nutrient Data Bank was used to analyze dietary intake in previous MND research studies (1985-1989). The USDA Standard Reference tapes and the Continuing Survey of Food Intakes by Individuals (CSFII) are the basis of the new CAN system database. The CAN system is different from other food analysis systems in that the USDA retention factors from CSFII can be applied to different cooking methods to estimate nutrient losses during cooking.

This manual documents the procedures that MND computer staff utilizes to analyze nutrient intakes from research studies. This technical report will be updated as major revisions to the different programs occur. Finally, the goal of this computerized analysis of nutrients documentation should not be lost in the complexity and cryptography of the language used to *describe the process:* to derive nutrient content from food intakes.

PREFACE

This document was written as a guide for users of the CAN system on the VAX computer. However, information on several pertinent features of the VAX was necessary to aid users in the operation of the CAN system. In this regard, USARIEM would like to acknowledge Digital Equipment Corporation (Burlington, Massachusetts) who granted us permission to print the material on VAX features presented in Section 5.1.2.1 of Chapter 5.

CHAPTER 1.0

NUTRITIONAL ANALYSIS SYSTEM FLOW CHART

Contributing Authors:
LTC Madeleine S. Rose
LTC Eldon W. Askew
Michael Benson

1.1 Introduction

The Military Nutrition Division assists the Surgeon General of the Army by monitoring the effects of the Army's rations and feeding programs on the health of the soldier. One of its major responsibilities is to determine if soldiers are consuming adequate quantities of foods to meet the Military Recommended Dietary Allowances (AR 40-25). Studies are conducted on soldiers eating and working in a garrison environment and in the field. Soldiers can be eating A, B, and/or T Rations in group feeding scenarios or eating Meal, Ready to Eat (MRE); Ration, Cold Weather (RCW); Ration, Lightweight (RLW), or experimental operational rations in individual feeding situations. Analyzing the consumption of operational rations is simple because data are available for all ration components. Approximately half of the research studies examine food consumption in garrison dining facilities. Analyzing garrison feeding is more complicated because the cooks have a larger variety of fresh, frozen, and canned food items at their disposal and a larger variety of cooking methods are available. Recipe Specialists from USARIEM observe recipe preparation to capture data on the amounts and types of ingredients that are used in recipe preparation. Obtaining the nutritional analyses of these different types of foods, collecting food consumption and recipe data in the dining hall or field, and analyzing this data is a complicated undertaking.

In order to analyze the data that were being generated by these research studies, the Computerized Analysis of Nutrients (CAN) system was developed. It includes: a nutrient database, recipe analysis component, diet history analysis program, software to enter food consumption and other biochemical data, and programs for combining food consumption and database information. The purpose of this report is to describe the component parts of this system and present the entire system in a user's guide format.

1.2 Flowcharts

Figure 1.1 displays the overall process from notification of the study through data collection to graphic display of the information. Figure 1.2 gives a detailed analysis of the whole process from a programmer's perspective.

Figure 1.1 Flow Chart of Overall CAN System

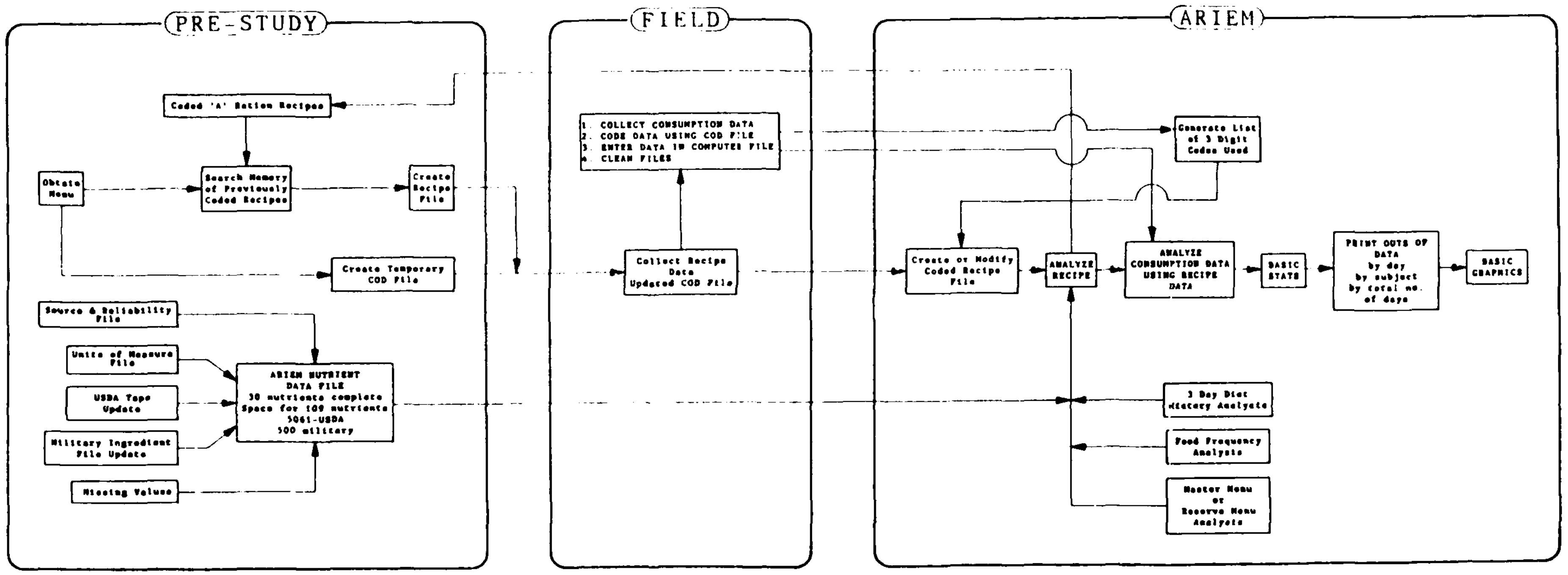

PRE-STUDY
Coded 'A' Ration Recipes
Obtain Menu
Search Memory of Previously Coded Recipes
Create Recipe File
Create Temporary COD File
Source & Reliability File
Units of Measure File
USDA Tape Update
Military Ingredient File Update
Missing Values
ARIEM NUTRIENT DATA FILE 30 nutrients complete Space for 109 nutrients 5061-USDA 500 military
FIELD
1. COLLECT CONSUMPTION DATA
2. CODE DATA USING COD FILE
3. ENTER DATA IN COMPUTER FILE
4. CLEAN FILES
Collect Recipe Data Updated COD File
ARIEM
Generate List of 3 Digit Codes Used
Create or Modify Coded Recipe File
ANALYZE RECIPE
ANALYZE CONSUMPTION DATA USING RECIPE DATA
BASIC STATS
PRINT OUTS OF DATA by day by subject by total no. of days
BASIC GRAPHICS
3 Day Diet History Analysis
Food Frequency Analysis
Master Menu or Reserve Menu Analysis

Figure 1.2 Programmer's Flow Chart of CAN System

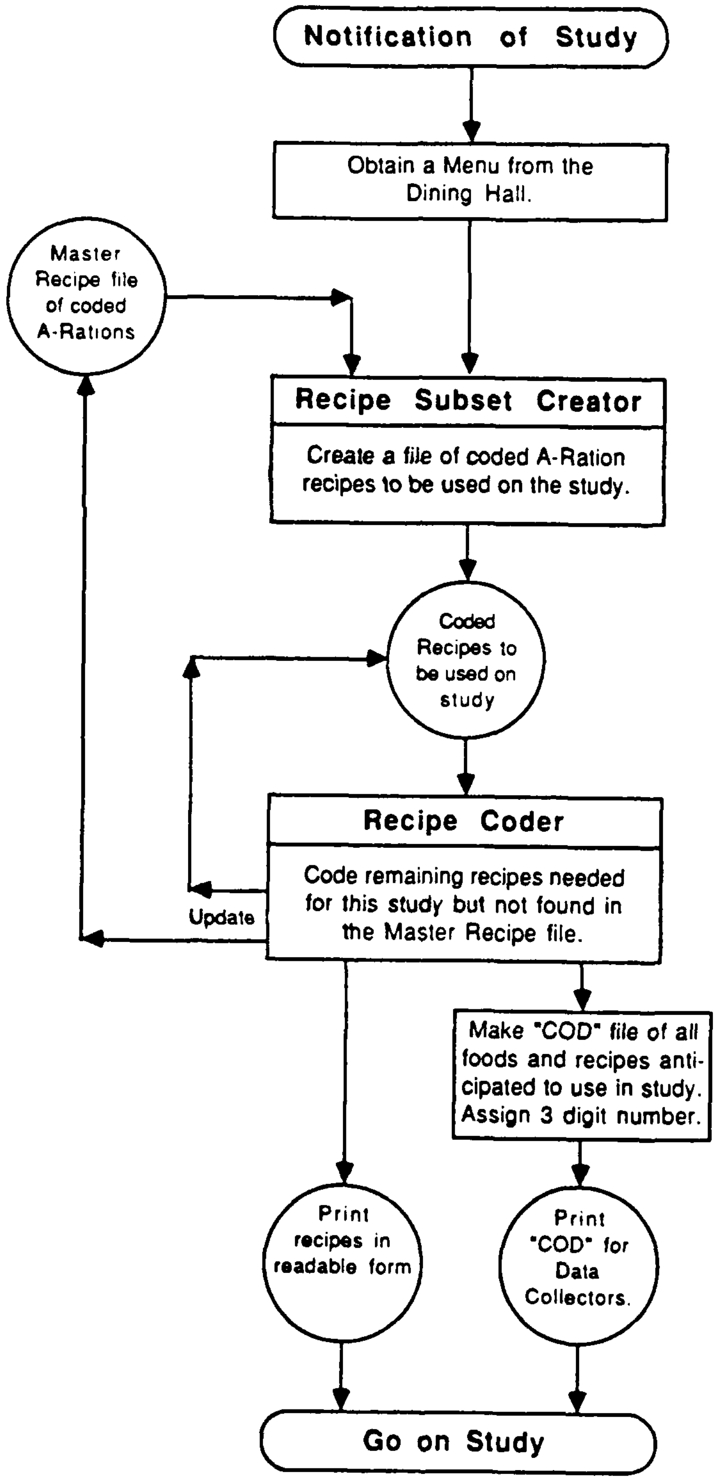

Notification of Study
Obtain a Menu from the Dining Hall.
Master Recipe file of coded A-Rations
Recipe Subset Creator
Create a file of coded A-Ration recipes to be used on the study.
Coded Recipes to be used on study
Recipe Coder
Code remaining recipes needed for this study but not found in the Master Recipe file.
Update
Make "COD" file of all foods and recipes anticipated to use in study. Assign 3 digit number.
Print recipes in readable form
Print "COD" for Data Collectors.
Go on Study

Figure 1.2 (Continued)

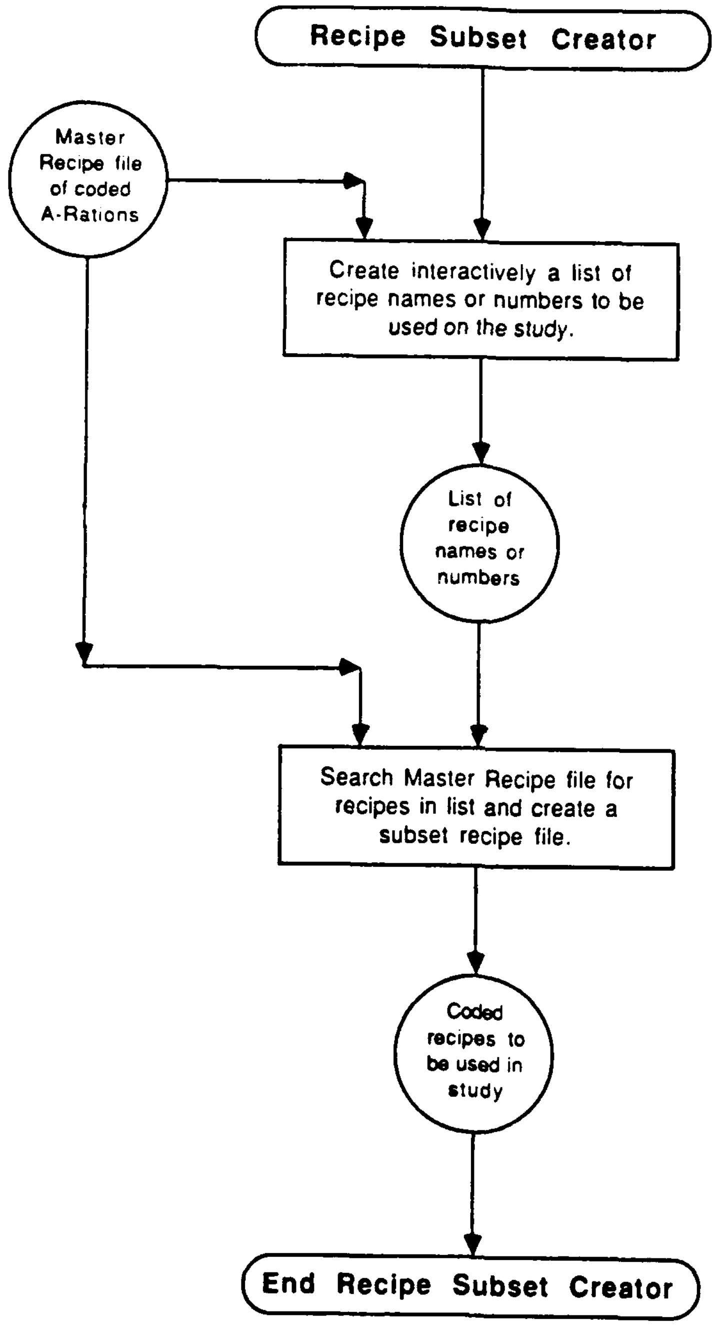

Recipe Subset Creator
Master Recipe file of coded A-Rations
Create interactively a list of recipe names or numbers to be used on the study.
List of recipe names or numbers
Search Master Recipe file for recipes in list and create a subset recipe file.
Coded recipes to be used in study
End Recipe Subset Creator

Figure 1.2 (Continued)

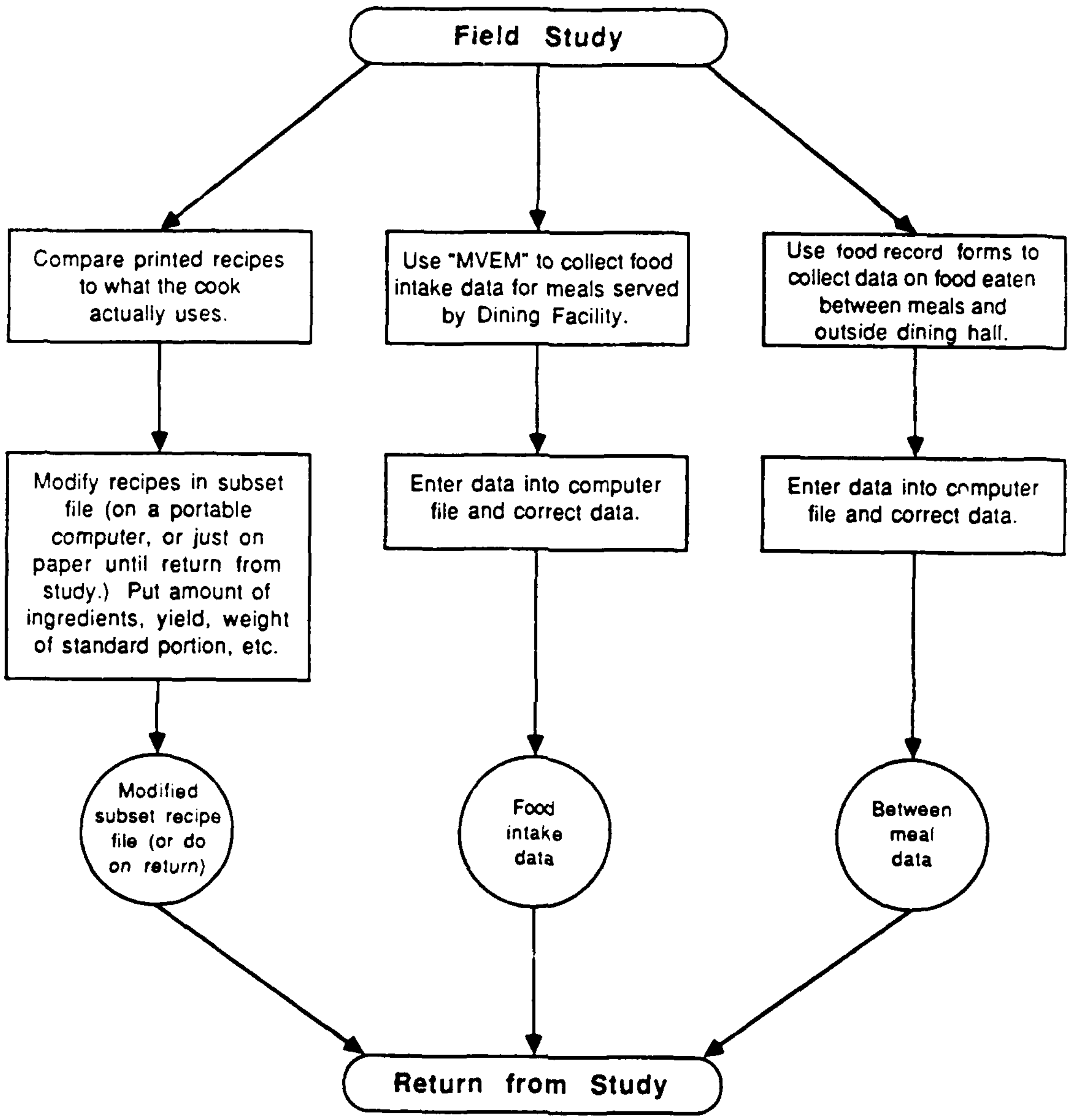
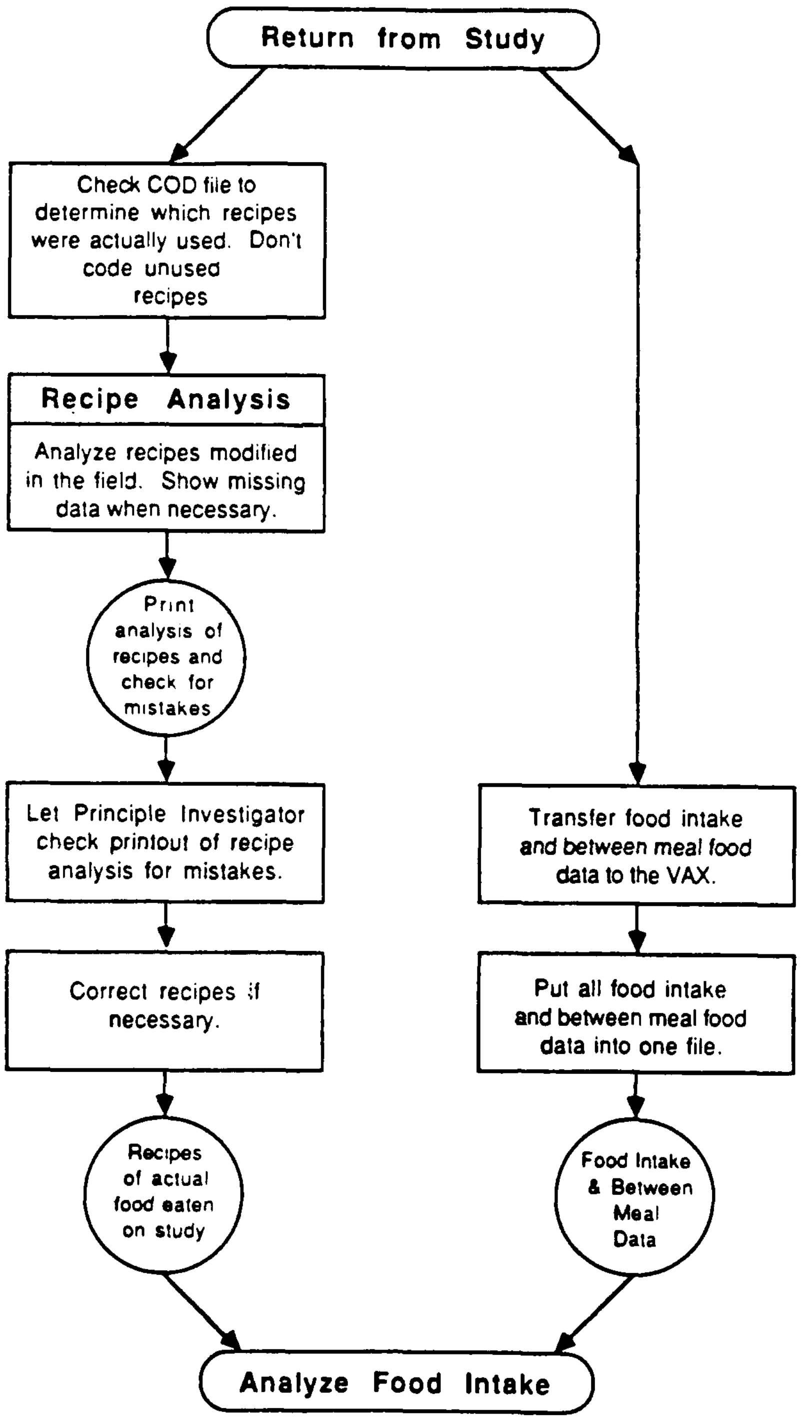
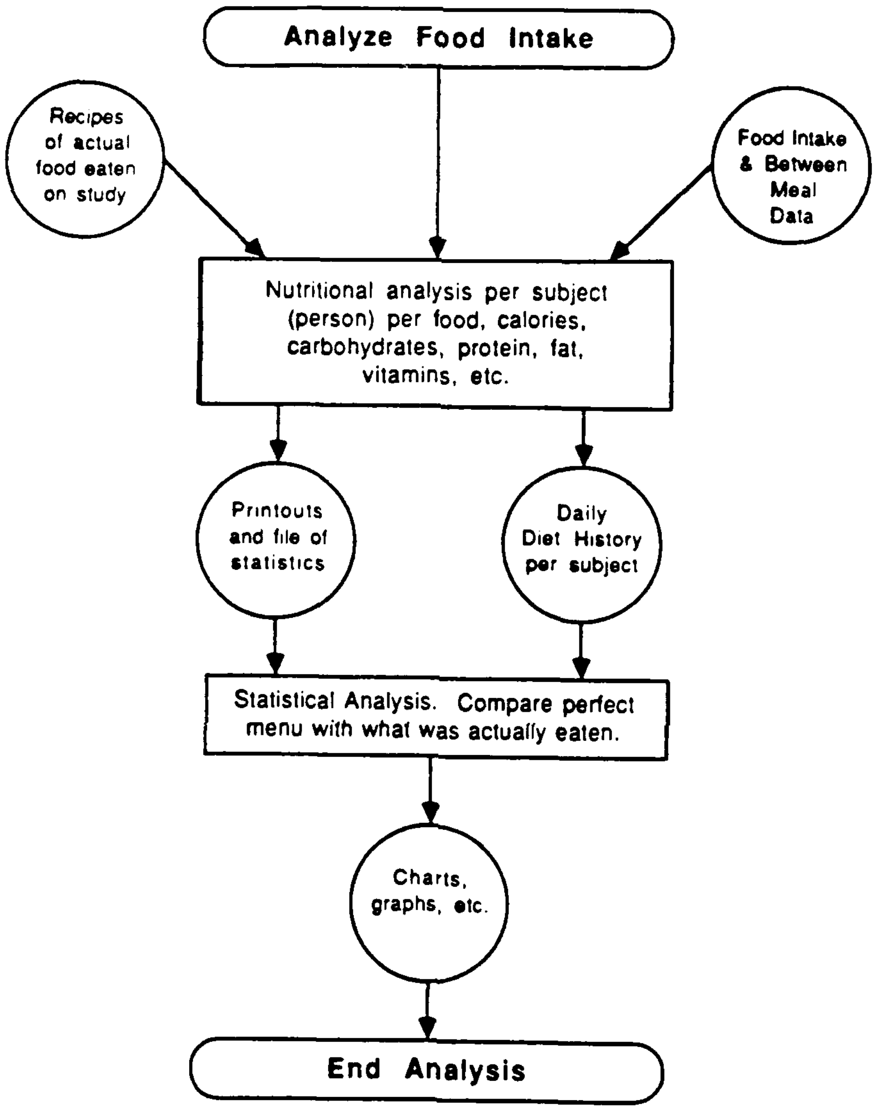

Field Study
Compare printed recipes to what the cook actually uses.
Use "MVEM" to collect food intake data for meals served by Dining Facility.
Use food record forms to collect data on food eaten between meals and outside dining hall.
Modify recipes in subset file (on a portable computer, or just on paper until return from study.) Put amount of ingredients, yield, weight of standard portion, etc.
Enter data into computer file and correct data.
Enter data into computer file and correct data.
Modified subset recipe file (or do on return)
Food intake data
Between meal data
Return from Study

Figure 1.2 (Continued)



Return from Study
Check COD file to determine which recipes were actually used. Don't code unused recipes
Recipe Analysis
Analyze recipes modified in the field. Show missing data when necessary.
Print analysis of recipes and check for mistakes
Let Principle Investigator check printout of recipe analysis for mistakes.
Correct recipes if necessary.
Recipes of actual food eaten on study
Transfer food intake and between meal food data to the VAX.
Put all food intake and between meal food data into one file.
Food Intake & Between Meal Data
Analyze Food Intake

Figure 1.2 (Continued)


Analyze Food Intake
Recipes of actual food eaten on study
Food Intake & Between Meal Data
Nutritional analysis per subject (person) per food, calories, carbohydrates, protein, fat, vitamins, etc.
Printouts and file of statistics
Daily Diet History per subject
Statistical Analysis. Compare perfect menu with what was actually eaten.
Charts, graphs, etc.
End Analysis

CHAPTER 2.0

NUTRIENT DATABASE

Contributing Authors:
Dr. Kenneth Samonds
Carlo Radovsky

2.1 Nutrient Database Flow Charts

Figure 2.1 contains information on all the programs and files that were used to generate the Nutrient Database from the USDA Standard Reference tapes and the Continuing Survey of Food Intakes by Individuals (CSFII).

2.2 Checklist of Features of Nutrient Database

A. Three sources of nutrient data
   1. USDA Standard Reference version 6
      a. information provided on 73 nutrients
      b. includes missing values
   2. Continuing Survey of Food Intakes by Individuals (CSFII) Primary (Ingredients) File
      a. information provided on 20 nutrients
      b. no missing values - all blanks filled with imputed values
   3. CSFII Master File
      a. includes fast foods and combination foods for diet history analysis
   4. Nutrient data from above 3 sources combined, duplicate data eliminated; Nutrient Database is not yet complete for 20 nutrients

B. Unique features of USDA Standard Reference
   1. Space included for 109 nutrients for future expansion.
   2. Names associated with foods are from the codebooks *from these files*, not the files themselves - includes up to 80 characters; limited and standardized abbreviations; includes alternate, multiple spelling, etc.
   3. Weights of units of measure from codebooks are incorporated into a separate file which includes household measures.
   4. Source and reliability - based on number of analyses, standard deviation of analyses, whether imputed, etc.
   5. Food Group categories
   6. Nutrient Retention factors which can be applied to the food groups during processing.

C. Organized in Oracle Tables that are interrelated.

D. On-screen editing of all Oracle Tables (FORMS) and review capabilities.

E. Future enhancements:
   1. Will include items unique to military recipes and rations, some Army sub-recipes, proprietary data, etc.
   2. Will supplement units table with drained weight yields, standard serving sizes.
   3. Cross reference with Federal Supply Catalog
   4. Will flag food items appropriate for A, B, and/or T rations.

2.3 Transfer of Military Ration Data From Food Engineering Directorate (FED)

Objective:
To transfer the nutrient data on military rations from the Food Engineering Directorate (FED) computer to the Oracle database of the CAN system at USARIEM.

Background:
Previously, nutritional data were received in hard copy format and re-entered at USARIEM. In 1989, a process was developed to transfer the data from the FED UNIVAC to computer tape and then to USARIEM's VAX 780.

Procedure:
The following is an outline of the steps required for the ration data to become accessible to MND software for analysis of consumption data:

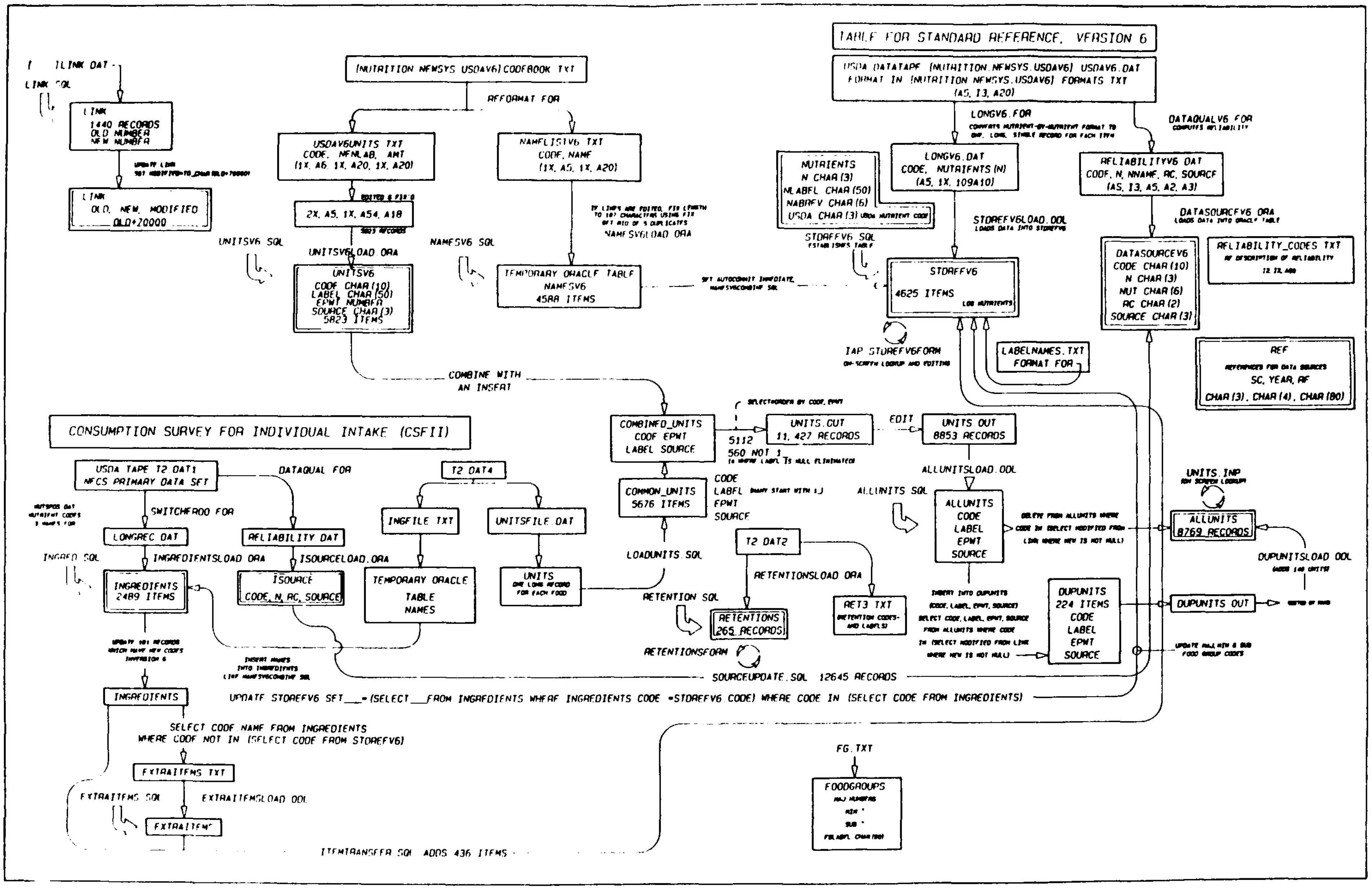
1. Obtain from FED two computer tapes containing all of their raw nutrient data (1 for fat and the other for proximates, vitamins, and minerals) and a hard copy description of the data stream.
2. Have the Information Management Branch load the computer tape into a subdirectory of the NUTRITION account on the VAX.
3. Run the Oracle Data Loader (ODL) program to place the information into a temporary table in Oracle. ODL requires that each record exist upon only one line. It may be necessary to change the format of the FED file to accommodate this requirement.
4. Once the data are loaded in Oracle, run a command file that extracts all occurrences of a single item, calculates means and standard deviations, and places the results into an Oracle table.
5. At this point spot check the data to confirm that the data were transferred and processed correctly.
6. Once you are confident that the data are correct, place the information in its appropriate study file. Decision has not been made as to how and where the data should be located.

Future Plans:
FED anticipates changing from UNIVAC to IBM about December 1989. The process for transfer from FED's new IBM will be very similar but will require some readjustments.

2.4 ORACLE Relational Database System

All data for the Nutrient Database are stored in the Oracle Relational Database Management System. This system was chosen because of its flexibility and capacity for handling large amounts of data. Being a relational database, Oracle allows the definition of complex relationships between data items, regardless of how or where the information is stored. Oracle also allows programming languages, such as FORTRAN and Pascal, to perform direct calls to the data, thus enabling complex interactive

Figure 2 1 Flowchart for USDA Data



LINK DAT
LINK SQL
LINK 1440 RECORDS OLD NUMBER NEW NUMBER
LINK OLD, NEW, MODIFIED OLD+20000
(NUTRITION NEWSYS USDAV6) CODEBOOK TXT
REFORMAT FOR
USDAV6UNITS TXT CODE, NEWLAB, AMT (1X, A6, 1X, A20, 1X, A20)
2X, A5, 1X, A54, A18
UNITSV6 SQL
UNITSV6LOAD ORA
UNITSV6 CODE CHAR (10) LABEL CHAR (50) EPWT NUMBER SOURCE CHAR (3) 5823 ITEMS
NAMELISTV6 TXT CODE, NAME (1X, A5, 1X, A20)
NAMESV6LOAD ORA
NAMESV6 SQL
TEMPORARY ORACLE TABLE NAMESV6 4588 ITEMS
COMBINE WITH AN INSERT
TABLE FOR STANDARD REFERENCE, VERSION 6
USDA DATATAPE (NUTRITION NEWSYS USDAV6) USDAV6 DAT FORMAT IN (NUTRITION NEWSYS USDAV6) FORMATS TXT (A5, I3, A20)
LONGV6 FOR
LONGV6 DAT CODE, NUTRIENTS (N) (A5, 1X, 109A10)
DATAQUALV6 FOR
RELIABILITYV6 DAT CODE, N, NNAME, RC, SOURCE (A5, I3, A5, A2, A3)
NUTRIENTS N CHAR (3) NLABEL CHAR (50) NABREV CHAR (6) USDA CHAR (3)
STOREFV6 SQL
STOREFV6LOAD ODL
STOREFV6 4625 ITEMS
IAP STOREFV6FORM
DATASOURCEV6 ORA
DATASOURCEV6 CODE CHAR (10) N CHAR (3) NUT CHAR (6) RC CHAR (2) SOURCE CHAR (3)
RELIABILITY_CODES TXT
LABELNAMES TXT FORMAT FOR
REF SC, YEAR, RF CHAR (3), CHAR (4), CHAR (80)
CONSUMPTION SURVEY FOR INDIVIDUAL INTAKE (CSFII)
USDA TAPE T2 DAT1 NFCS PRIMARY DATA SET
DATAQUAL FOR
SWITCHEROO FOR
LONGREC DAT
RELIABILITY DAT
INGRED SQL
INGREDIENTSLOAD ORA
ISOURCELOAD ORA
INGREDIENTS 2489 ITEMS
ISOURCE CODE, N, RC, SOURCE
T2 DAT4
INGFILE TXT
UNITSFILE DAT
TEMPORARY ORACLE TABLE NAMES
UNITS
COMBINED_UNITS CODE EPWT LABEL SOURCE
COMMON_UNITS 5676 ITEMS
LOADUNITS SQL
CODE LABEL EPWT SOURCE
5112
560 NOT 1
UNITS CUT 11, 427 RECORDS
EDIT
UNITS OUT 8853 RECORDS
ALLUNITSLOAD ODL
ALLUNITS SQL
ALLUNITS CODE LABEL EPWT SOURCE
UNITS INP
ALLUNITS 8769 RECORDS
DUPUNITSLOAD ODL
DUPUNITS 224 ITEMS CODE LABEL EPWT SOURCE
DUPUNITS OUT
T2 DAT2
RETENTIONSLOAD ORA
RETENTION SQL
RETENTIONS 265 RECORDS
RET3 TXT
RETENTIONSFORM
SOURCEUPDATE SQL 12645 RECORDS
INGREDIENTS
UPDATE STOREFV6 SET ___ = (SELECT ___ FROM INGREDIENTS WHERE INGREDIENTS CODE = STOREFV6 CODE) WHERE CODE IN (SELECT CODE FROM INGREDIENTS)
SELECT CODE NAME FROM INGREDIENTS WHERE CODE NOT IN (SELECT CODE FROM STOREFV6)
EXTRAITEMS TXT
EXTRAITEMS SQL
EXTRAITEMSLOAD ODL
EXTRAITEMS
ITEMTRANSFER SQL ADDS 436 ITEMS
FG TXT
FOODGROUPS

manipulations of the data. As Oracle runs on a Digital VAX 11/780 computer, a great amount of data can be stored and accessed. Currently over one million values are stored in our database.

CHAPTER 3.0

RECIPE ANALYSIS

Contributing Authors:
John Finn
Michael Benson

3.1 Nutrition Project Report

3.1.1 Introduction

The Nutrition project at USARIEM consisted of the development of software to create and manipulate recipes. These recipes can then be analyzed for nutritional content by interaction with a large nutrient database.

Food consumption on individual subjects, and variations from the standard recipes during food preparation are recorded in the field. The food consumption data are entered into computer files in the field, and recipe data are brought back to USARIEM to create recipes using the Recipe program. The recipes are analyzed for nutrient content, and this data is combined with the food consumption data to produce information on the nutrients consumed by individual subjects.

3.1.2 Software Development

Until recently, most of the recipe manipulation and analysis had been done remotely on the University of Massachusetts at Amherst, MA (UMASS) computer. Software for manipulating and analyzing recipes existed at USARIEM, however it was never used because it was incomplete and inaccurate.

GEO-CENTERS, INC. of Newton Center, MA was contracted to assist in software preparation and debugging to achieve the goal of a self-contained system at our Institute that was not dependent on the UMASS computer.

The recipe coder and editor have been almost completely re-written. This was done to make the program easier to use, to fix existing bugs, and to add many necessary enhancements. The program is now completely menu driven and contains on-line help at every menu. It is also more modular and better documented for easier maintainability.

The recipe analysis program had many bugs which could cause inaccurate results. These bugs have been fixed, and the program has been integrated into the recipe coder and editor to create one coherent menu driven program.

The program for maintaining Fortran file versions of the Oracle database was updated to account for new Oracle table formats.

Finally, everything was well documented for the users as well as the programmers who will have to maintain and modify the programs in the future.

3.1.3 Documentation

Each of the major areas of the project that the contractor has been involved with has been documented in detail. There are currently four documents which are included with this report.

The first is the Recipe Coder / Editor / Analyzer User's Guide. This document explains how to run the software, how to create and modify recipes, and how to analyze them. It shows in a step by step manner what each menu looks like and how to use each of the options. It also contains a complete listing of all of the nutrients in the database along with their numbers and abbreviations.

The second document is the Recipe Coder / Editor / Analyzer Programmer's Guide. This document is primarily aimed at the programmers who will have to maintain and modify the program in the future, but it also contains useful information that anyone may need to know. It explains the formats of all the files used, shows the overall program structure, and explains the formulas used in the recipe analysis.

The third document explains the Oracle Table Translator program called Dumptable. This program converts the Oracle tables into Fortran files to be used with the Recipe and Diet History programs. The document explains how to use the program and shows the current Oracle table formats. It also shows the overall structure of the program.

The fourth document is the Menu Manager Programmer's Guide. The Menu Manager is a set of subroutines that are used by the other programs to handle menus and screen forms. The document explains in detail how these menus can be integrated into other programs so that any future programs can have a consistent "look and feel" with the current ones. It explains how to set up menus, and how to write programs that use the Menu Manager.

3.2 Recipe Coder / Editor / Analyzer User's Guide

3.2.1 Introduction

The Recipe Coder / Editor / Analyzer is an interactive program on the VAX for creating and editing coded recipe files and then analyzing them for nutritional composition. This document describes how to use the Recipe Coder / Editor / Analyzer. Currently, it should be considered a "living" document. As the program is updated, this document will also be updated.

3.2.2 Getting Started

To run the Recipe Coder / Editor / Analyzer (CEA), several files are needed.

All files, with the possible exception of the database files, must be in the same subdirectory. Currently, the format and the content of the database files are being updated, so this will eventually change.

The source code files (ending with .FOR) and the object files (ending with .OBJ) are not needed to run the program. The executable file (ending with .EXE) is the actual program. There are also several "resource" files (ending with .RSC). These files contain templates for the menus and are needed to run the program. Help files (ending with .HLP) are also used by the program for on-line help. Note that for every .RSC file there is a .HLP file. The following tables show the files needed to run the Recipe Coder / Editor:

Database Files	File Name Description
INGREDV6.DAT	Ingredients data file.
UNITSV6.DAT	Units of measure.
GROUPRET.DAT	Food group retention factors.
RETENTIONS.DAT	Ingredient retention factors.

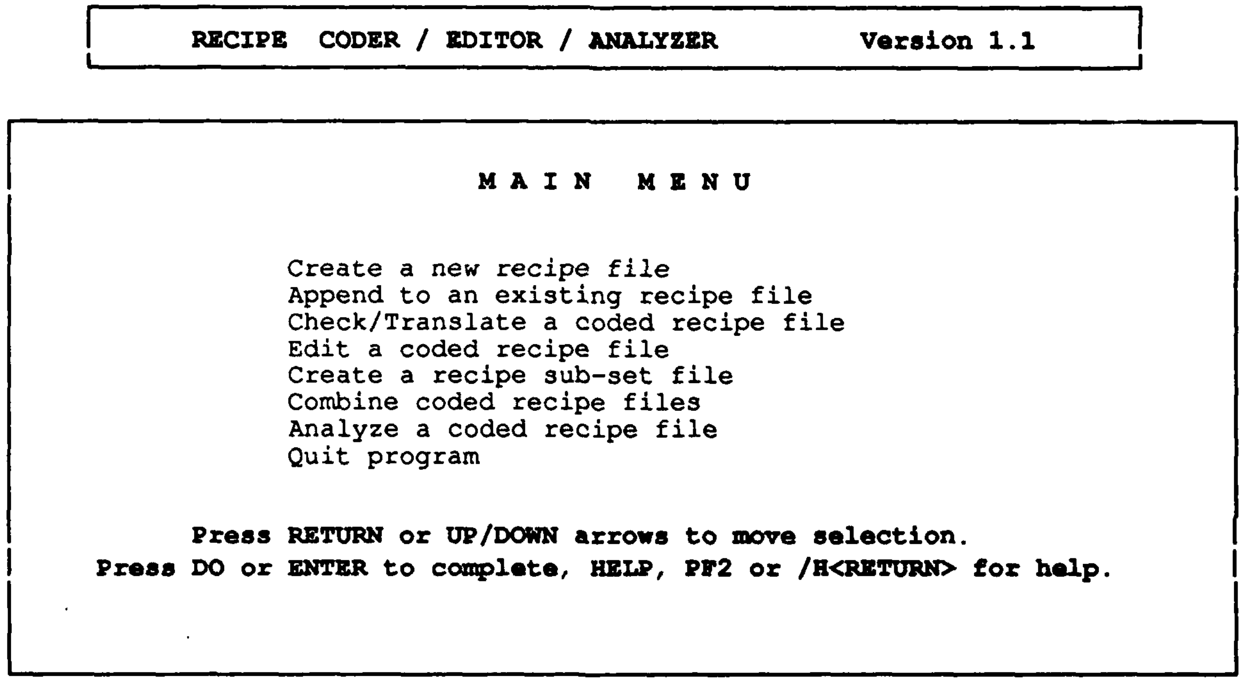
Program Files:	File Name Description
RECIPE.EXE	The main executable program.
RECMENU1.RSC	The main menu template.
RECMENU1.HLP	The main menu help.
RECMENU2.RSC	Recipe naming menu template.
RECMENU2.HLP	Recipe naming help.
RECMENU3.RSC	Recipe entry menu template.
RECMENU3.HLP	Recipe entry help.
RECMMS.RSC	Food group selection menu template.
RECMMS.HLP	Food group selection help.
RECEDMENU1.RSC	Main editing menu template.
RECEDMENU1.HLP	Main editing help.
RECEDMENU2.RSC	Recipe duplication menu template.
RECEDMENU2.HLP	Recipe duplication help.
RECEDMENU3.RSC	Recipe line editing menu template.
RECEDMENU3.HLP	Recipe line editing help.
RECEDMENU4.RSC	Recipe header editing menu template.
RECEDMENU4.HLP	Recipe header editing help.
RECEDMENU5.RSC	Ingredient editing menu template.
RECEDMENU5.HLP	Ingredient editing help.
RECEDMENU6.RSC	Yield changes editing menu template.
RECEDMENU6.HLP	Yield changes help.
RECANA.RSC	The recipe analysis menu.
RECANA.HLP	Help for recipe analysis.
MMS.DAT	Food groups for menu selections.

To start the program, type "RUN RECIPE" at the DCL prompt. When the program starts, a header with the version number will be displayed followed by a message indicating that the program is performing initialization. This initialization will take about 2 or 3 minutes. When the initialization is complete, the main menu will appear.

3.2.3 The Main Menu

The main menu currently contains 8 items. To make selections, use the cursor keys or the RETURN key to move the selection bar up or down. Then, press ENTER on the numeric keypad to make the selection. Note that the selection bar wraps around from top to bottom and vice versa. The main menu is depicted in figure 3.1.

Figure 3.1 Main Menu



```
     RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

                          M A I N   M E N U

                   Create a new recipe file
                   Append to an existing recipe file
                   Check/Translate a coded recipe file
                   Edit a coded recipe file
                   Create a recipe sub-set file
                   Combine coded recipe files
                   Analyze a coded recipe file
                   Quit program

         Press RETURN or UP/DOWN arrows to move selection.
     Press DO or ENTER to complete, HELP, PF2 or /H<RETURN> for help.
```

To get on-line help at any menu, press PF2. If you are using a VT-220 compatible terminal, or higher, you can also use the DO key instead of ENTER, and the HELP key instead of PF2. However, if you are using a terminal emulator on a DEC PC or other PC, the HELP and DO keys have different meanings.

The first item in the Main Menu is "Create a new recipe file". Select this to create a new recipe file from scratch. The next item is "Append to an existing recipe file". Select this option to add recipes to an existing coded recipe file. Regardless of which of those two items is selected, the actual coding process is identical.

The third item in the Main Menu is "Check/Translate a coded recipe file". Select this option to produce an index file and optional printout of a coded recipe file in a readable format. An index file is a listing of the recipe codes and names in a recipe file with or without a description.

The fourth item in the Main Menu is "Edit a coded recipe file". This selection will bring up a series of other menus for editing existing recipes line by line.

The fifth item in the Main Menu is "Create a recipe sub-set file". This can be used to create a new file that contains a sub-set of the coded recipes in another file. For example, suppose you have a very large coded recipe file with all of the A-rations. You might want to create a sub-set file containing only the recipes you will be using on a particular survey.

The sixth item in the Main Menu is "Combine coded recipe files". This is the opposite of the above item. For example, if you have several small coded recipe files, you can use this option to combine them into one larger file.

The seventh item in the Main Menu is "Analyze a coded recipe file". This will bring up a menu for recipe analysis. A report of the analysis can be printed without quitting from the program.

The last item in the Main Menu is "Quit program". Select this when you are completely finished. Since it takes 2 or 3 minutes to reinitialize, the program gives you a chance to change your mind before terminating.

3.2.3.1 Creating and Appending to Recipe Files

After "Create" or "Append" is selected from the Main Menu, you will be prompted to open the recipe file. Type a name for the new file, or the name of an existing file into the form shown in figure 3.2.

Figure 3.2 Type File Name Menu

```
RECIPE CODER / EDITOR / ANALYZER          Version 1.1

Type file name, RETURN, or PF4 to cancel:        junk.rec
```

In this example, the user typed "junk.rec". If the file does not open properly, the user is prompted to retry, or cancel the operation and return to the Main Menu. If this is not the first file that has been opened in the current session, the previous file will appear as the default choice. You can select that file by pressing RETURN rather than typing out the whole name. The default name menu is shown in figure 3.2.1.

Figure 3.2.1 Type File Name/Default Name Menu

```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

   Type file name, RETURN, or PF4 to cancel:        junk.rec
   <RETURN> for default:  JUNK.REC
```

After the file is opened, the Recipe Naming Menu appears. This menu is used for naming the recipe and giving it a code number. Like other menus, use the cursor keys or RETURN to move the cursor from one field to another. Then, type into the field and press RETURN. Note: the contents of a menu field cannot be edited; it must be retyped. Figure 3.3 is what the Recipe Naming Menu looks like when it displays.

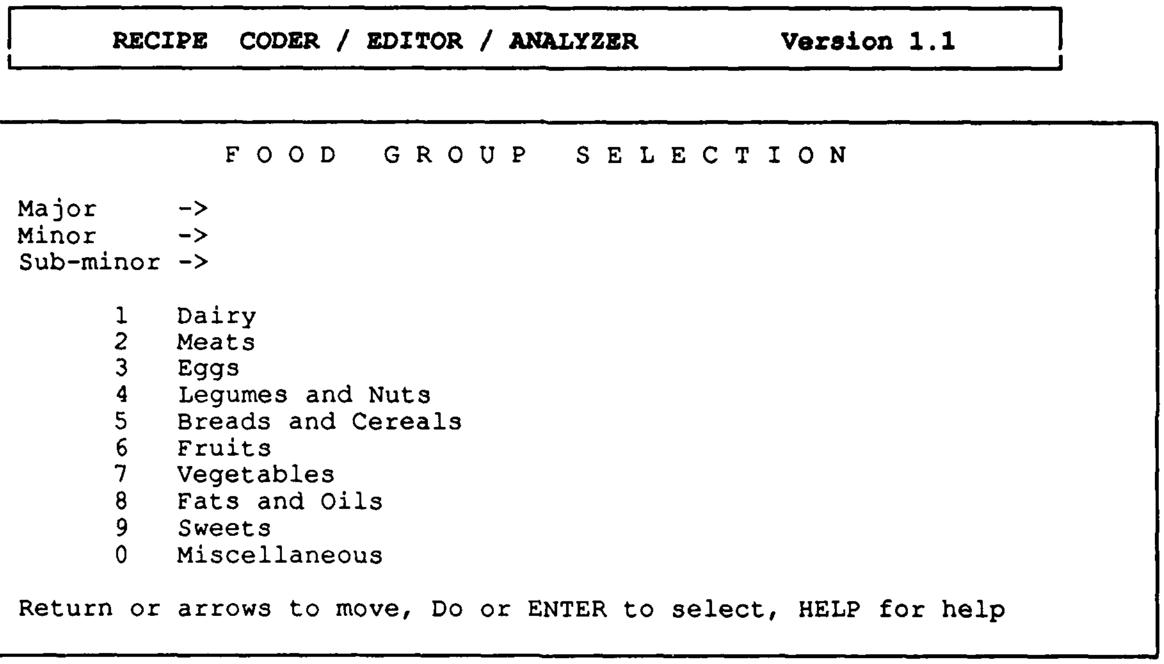
Figure 3.3 Recipe Naming Menu

```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

   Current file:       JUNK.REC

   Recipe name:    RECIPE

   Recipe Code:    RECCODE001


         Press RETURN or arrows to move selection, PF4 to cancel.
   Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.



   Name of recipe in file.  There can be many recipes in a file.
```

Notice that at the bottom of the menu there is a note line. This line shows information associated with the field the cursor is currently on. (The cursor will be blinking on the screen.) Also, like other menus, on-line help can be displayed by pressing PF2 (or HELP on some terminals).

When you are satisfied with the recipe name and code, press ENTER (or DO on some terminals) to complete the menu. The next menu to appear will then be the Food Group Selection Menu, shown in figure 3.4.

Figure 3.4 Food Group Selection Menu



```
RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

                 F O O D   G R O U P   S E L E C T I O N

     Major     ->
     Minor     ->
     Sub-minor ->

           1   Dairy
           2   Meats
           3   Eggs
           4   Legumes and Nuts
           5   Breads and Cereals
           6   Fruits
           7   Vegetables
           8   Fats and Oils
           9   Sweets
           0   Miscellaneous

     Return or arrows to move, Do or ENTER to select, HELP for help
```

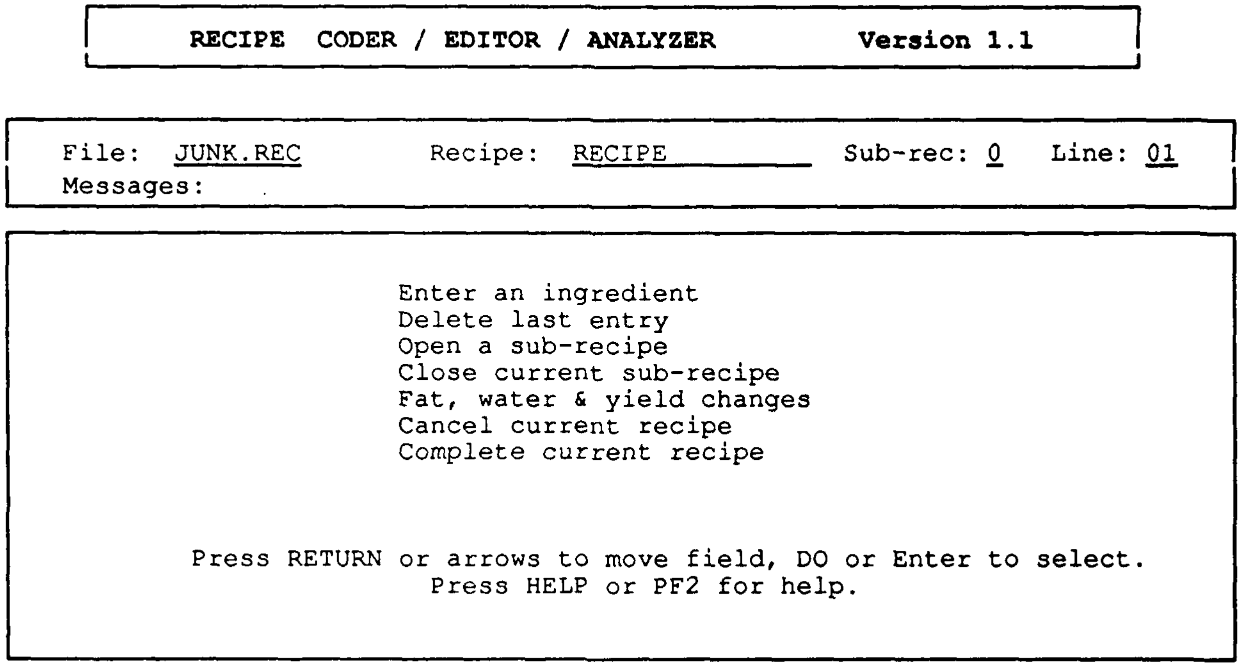
The first time this menu appears, the "Major" field will be blinking (on some terminals it may appear bolded). This indicates that you are selecting a major food group. Designating food groups at this point simplifies the analysis of food group contributions to different nutrients. Like the other menus, move the selection bar up and down with the arrow keys (or RETURN). Then make the selection by pressing ENTER (or DO on some terminals).

After you make the major food group selection, the menu will be redrawn and the "Minor" field will be blinking. This indicates that you are selecting a minor food group. Select a minor food group the same way as above.

The third time the menu is drawn, the "Sub-minor" field will be blinking to indicate you are selecting a sub-minor food group. After you make a selection, a message at the bottom of the menu will prompt you to either press ENTER if you are satisfied, or PF4 to start the food group selection process over. At any point in any of the three passes through the menu, you can press PF4 to start over. If you choose a food group which for some reason is not in the current database, the message at the bottom of the menu will indicate this and prompt you to try again.

When the food group has been selected successfully, the Recipe Entry Menu will appear as shown in Figure 3.5.

Figure 3.5 Recipe Entry Menu



```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

   File:  JUNK.REC        Recipe:  RECIPE          Sub-rec: 0   Line: 01
   Messages:

                          Enter an ingredient
                          Delete last entry
                          Open a sub-recipe
                          Close current sub-recipe
                          Fat, water & yield changes
                          Cancel current recipe
                          Complete current recipe


             Press RETURN or arrows to move field, DO or Enter to select.
                          Press HELP or PF2 for help.
```

Notice that below the screen header there is a status box. This box shows the current recipe file, the recipe within that file, the number of sub-recipes currently open, and the current line number within the recipe. There is also a line for messages which appear from time to time. This box will remain on the screen throughout the recipe coding process.

The first choice in the menu is "Enter an ingredient". When this item is selected, the menu will temporarily disappear and you are prompted to enter the name of the ingredient. You can enter a partial name, the full name, or if the ingredient code is known, a backslash, "/", and the food code. Upper and lower case does not matter. After you type the ingredient name followed by RETURN, a list of all of the possible ingredients containing the name entered will appear. For example, if you enter "sugar", you will get a list of all of the different ingredients that have "sugar" in their names such as "brown sugar", "maple sugar", "sugar substitute", etc. These will all be numbered. You can then choose the number that corresponds to the exact ingredient you want. If there are more ingredients than can fit on the screen, the prompt at the bottom of the screen will indicate that you can press RETURN to see more choices. Some ingredient names are identical when viewed in sixty character format. To see an ingredient's eighty character name description, enter the ingredient number and then press PF3. Press PF1 to restart from the beginning of the list. Pressing PF4 will cancel the operation and return the user to the Recipe Entry Menu (Figure 3.5).

After you have chosen an ingredient, you will be prompted to choose the unit of measure being used such as pounds, grams, cups, etc. The number of choices will depend on the individual ingredient. Then, you are prompted to enter the amount of the ingredient in the chosen units. For example, you will be asked something like, "How many cups of brown sugar do you want?" You then enter the number (decimal allowed) and press RETURN.

Depending on the individual ingredient, the list of processing codes may appear at this point in the program. Processing codes adjust the vitamin and mineral retention values depending on anticipated losses from the cooking methods. The processing codes do not adjust for water, fat, or yield weight changes which would occur during the cooking process. Whether there are any processing codes or not for the ingredient, you will be able to enter a specific code rather than choosing one from the list in the future. But, you must be careful to choose a processing code that makes sense if you do not choose one from the list. If no processing codes are desired, press PF4 to exit from the Processing Code Entry Menu. You can choose up to 3 processing codes, one at a time, for each ingredient. If fewer than 3 codes are desired, press PF4 after entering the last one.

After all of the ingredient choices are made, the Recipe Entry Menu will reappear (Figure 3.5). At this point, if you made a mistake, you could select "Delete last entry" to delete the last ingredient you entered. Each time this option is selected, the last line in the recipe will be deleted and the status box will be updated accordingly.

To open a new sub-recipe, select "Open a sub-recipe". This will update the sub-recipe number in the status box. Up to 9 levels of sub-recipes may be nested. When a recipe is complete, all pending sub-recipes will be closed. It is only necessary to select "Close current sub-recipe" if you want other sub-recipes or ingredients after it. When you complete the recipe, all pending sub-recipes are closed automatically. Sub-recipes allow separate water and fat adjustments for different parts of a recipe. Adjustments to the final fat or water percentage are only processed for the total recipe, not for individual sub-recipes.

"Fat, water & yield changes" can be selected to insert a recipe modifier such as fat loss / gain, water loss / gain. Depending on what you choose, you may be prompted for other information. The list of choices will appear as in figure 3.6.

Figure 3.6 Ingredient Modifier Menu

```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

   File:  JUNK.REC        Recipe:  RECIPE          Sub-rec: 0   Line: 02
   Messages:
Choose an ingredient modifier:

  1:. F*:  change in fat expressed as % of initial recipe weight
  2: FR:   change in fat expressed as % of initial fat content
  3: F%:   final percentage of recipe which is fat
  4: W*:   change in water expressed as % of initial weight
  5: WR:   change in water expressed as % of initial water content
  6: W%:   final percentage of recipe which is water
  7: SS:   serving size in grams
  8: # :   number of servings in recipe
  9: Y :   final weight expressed as % of initial weight


 Type choice from 1 to   9 or
   Pf4 to cancel or PF1 to restart list:
```

After selecting the ingredient modifier, the Recipe Entry Menu will reappear (Figure 3.5). When you are finished with the recipe, you can select "Complete current recipe" to close pending sub-recipes and write out the information to the file, or "Cancel current recipe" to cancel the whole recipe entry session. If you choose to cancel, you will be given a second chance to change your mind.

When the recipe is completed or canceled, the Recipe Naming Menu reappears (Figure 3.3). This enables you to start a new recipe in the same file. Change the recipe name and recipe code, then press ENTER to start a new recipe entry session. If you are done with the whole recipe file, press PF4 to get back to the Main Menu.

3.2.3.2 Checking a Coded Recipe File

Once a coded recipe file has been created, you may wish to check it. You can make a file and optional printout of a coded recipe file or create an index of the recipes in a recipe file by selecting "Check/Translate a coded recipe file" from the Main Menu. This will bring up the dialog box in Figure 3.7.

The translated recipe file will have the same name as the coded recipe file except it will have the .CHK extension. In the above example, the translated file would be named "junk.chk". If you just press RETURN on the second question, "no printout" will be assumed. When the translation is complete, the Main Menu will reappear.

Figure 3.7 Check File Menu

```
RECIPE  CODER / EDITOR / ANALYZER          Version 1.1
```

```
Type file name, RETURN, or PF4 to cancel:   junk.rec
in addition to the file, do you want a printout? (Y/N[def])  n
Translation in progress ...
```

3.2.3.3 Editing a Recipe

Once a coded recipe file has been created, you can edit any of the recipes in the file in several ways. It is helpful to have a printout of the translated recipes handy (see section 3.2.3.2) during an editing session. After you select "Edit a coded recipe file" from the Main Menu, you will be prompted to open a file and choose a recipe in that file for editing. If you do not know the code of the recipe to be edited, type a "?" followed by RETURN. This will display an index of the recipe codes and the names contained in the specified recipe file. You can now select which recipe to edit by entering the number that corresponds to the recipe that you want to edit. Then, the Recipe Editor Menu as depicted in figure 3.8 displays.

Figure 3.8 Recipe Editor Menu

```
RECIPE  CODER / EDITOR / ANALYZER          Version 1.1
```

```
File:  JUNK.REC          Recipe:  RECCODE001                 Line: 00
Messages:
```

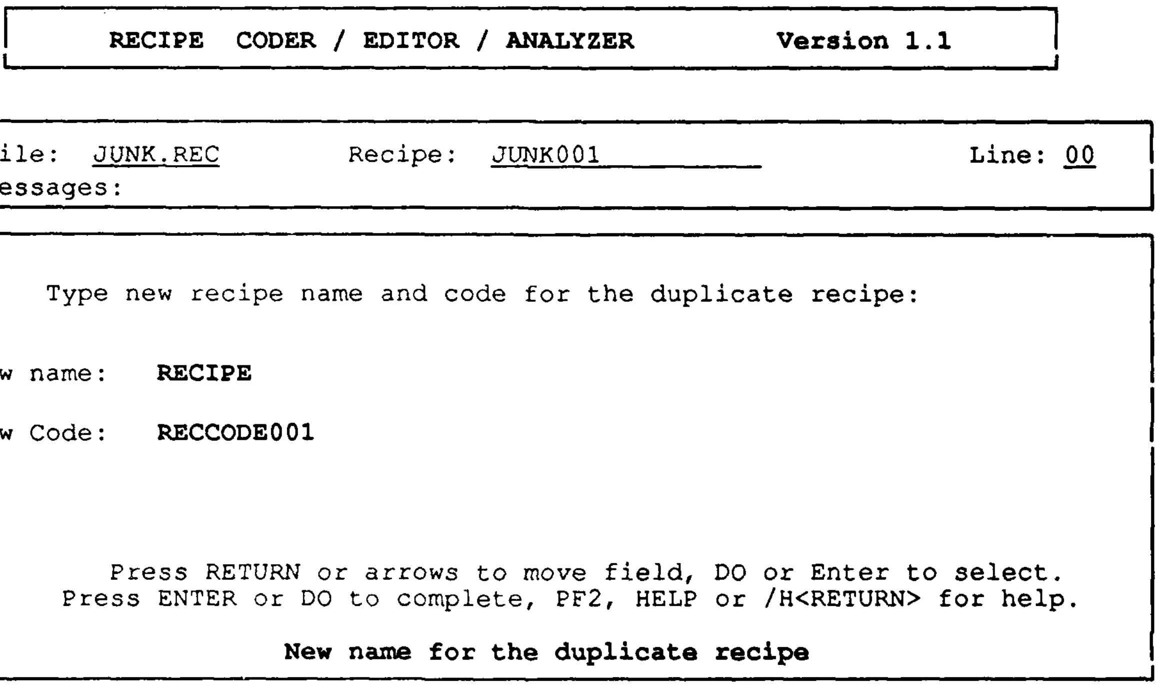
```
                    R E C I P E   E D I T O R

                      Edit this recipe
                      Delete this recipe
                      Duplicate this recipe
                      Exit recipe editor


        Press RETURN or arrows to move field, DO or Enter to select.
     Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.
```

The first choice in the menu is "Edit this recipe". This is for editing the recipe on a line by line basis. More on that later.

The second choice in the menu is "Delete this recipe". When you select this, you are given a second chance to change your mind before the recipe is actually deleted from the file.

"Duplicate this recipe" can be selected if you want to make a copy of a recipe. For example, suppose you have a recipe for tomato sauce and you want to make another recipe that is very similar, but has basil instead of oregano. You could first duplicate the recipe under a new name and then edit the new duplicate. The original version will be left intact and the duplicate version will have the changes. When you select "Duplicate this recipe", the Duplicate Recipe Naming Menu (Figure 3.9) appears to let you give the duplicate recipe a new name and code. The recipe name can remain the same although it is recommended that some unique descriptor be added to be able to distinguish between the two recipes. The recipe code must be changed to a unique code for that recipe file.

Figure 3.9 Duplicate Recipe Naming Menu



```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

   File:  JUNK.REC          Recipe:  JUNK001                      Line: 00
   Messages:

        Type new recipe name and code for the duplicate recipe:

  New name:   RECIPE

  New Code:   RECCODE001

            Press RETURN or arrows to move field, DO or Enter to select.
         Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.

                        New name for the duplicate recipe
```

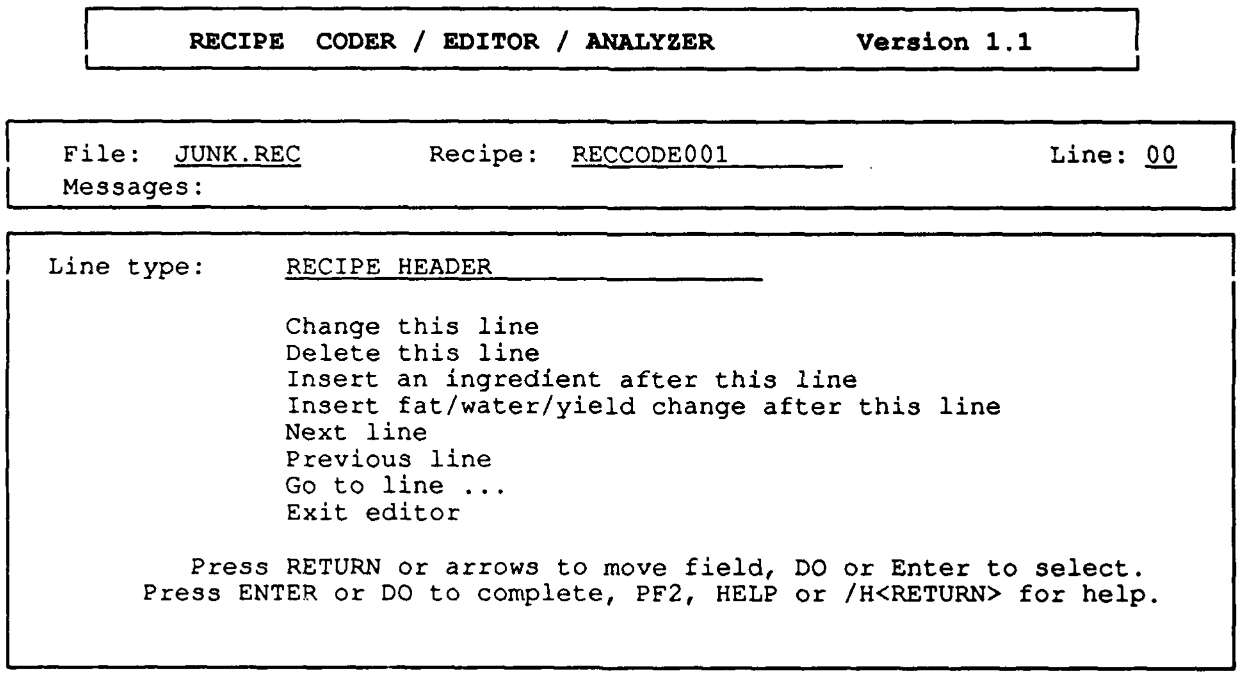
After you type the new name and code, press **ENTER** (or DO on some terminals). The recipe will then be duplicated under the new name and the status box will be updated.

The final selection on the Recipe Editor Menu is "Exit recipe editor". Selecting this option will return to the Main Menu.

3.2.3.3.1 Line by Line Recipe Editing

Recipes can be edited on a line by line basis to insert and delete ingredients and yield changes, or to change a particular ingredient, amount, unit of measure, or yield. Selecting "Edit this recipe" from the Recipe Editor Menu (Figure 3.8) brings up the Recipe Line Editor Menu shown in figure 3.10.

Figure 3.10 Recipe Line Editor Menu



```
        RECIPE  CODER / EDITOR / ANALYZER        Version 1.1

   File:  JUNK.REC        Recipe:  RECCODE001               Line: 00
   Messages:

  Line type:      RECIPE HEADER

                  Change this line
                  Delete this line
                  Insert an ingredient after this line
                  Insert fat/water/yield change after this line
                  Next line
                  Previous line
                  Go to line ...
                  Exit editor

            Press RETURN or arrows to move field, DO or Enter to select.
       Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.
```

At the top of the menu box is a field labeled "Line type". This shows what kind of line is in the recipe at the current line indicated in the status box. Line 0 is always the "recipe header". This contains the recipe name and code. Other line types may be "sub-recipe start / end", and "fat / water / yield change". If the line contains an ingredient code, the name of the ingredient will appear in the line type field. The "sub-recipe start / end" lines cannot be edited since they must be properly nested.

You can move through the recipe to find a specific line in three ways. First, you can advance through the recipe line by line by selecting "Next line" from the Recipe Line Editor Menu. Move the selection bar down to that command by using the down arrow key or the RETURN key. Then, press ENTER (or DO on some terminals) to advance one line. Notice that the line count in the status box and the line type in the menu box are updated. You can continue to press ENTER until you reach the line you are looking for.

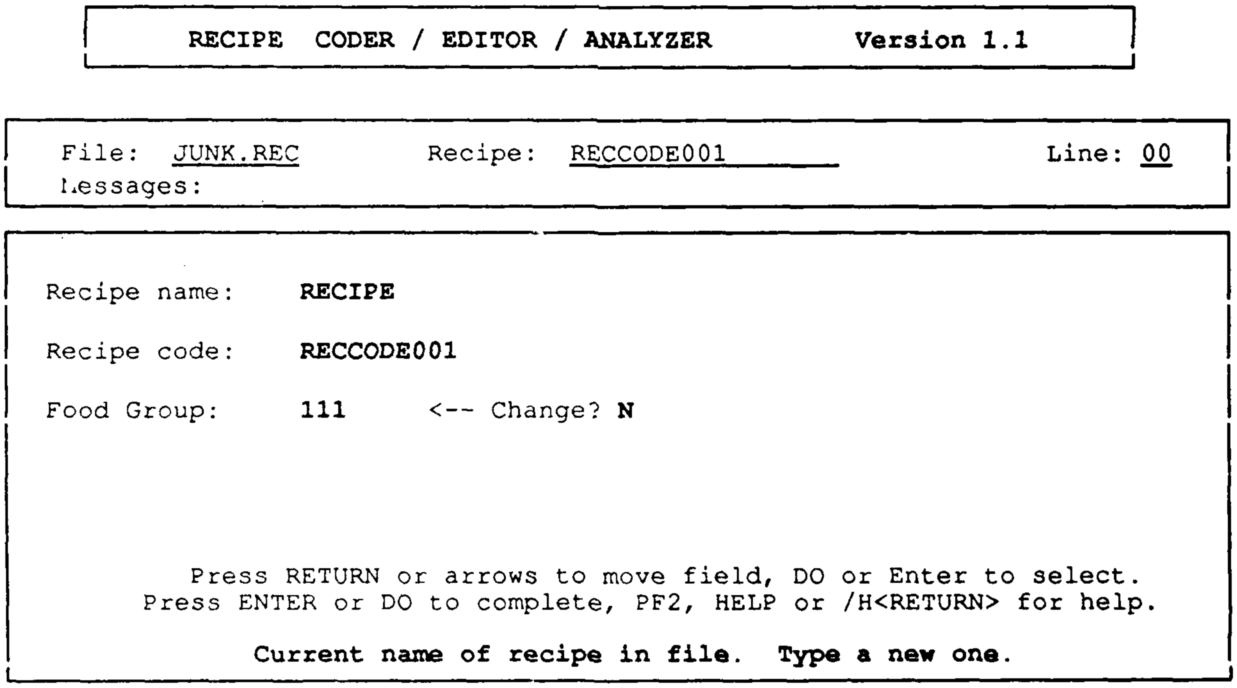
Secondly, you can back up through the recipe by selecting "Previous line" from the Recipe Line Editor Menu. This is done in the same way as with "Next line".

The third way to move through a recipe is to go directly to a specific line. If you have a checked listing handy (see section 3.2.3.2), you may know exactly where you want to go, so select "Go to line _" from the menu. At the bottom of the menu, you will be prompted to type the line number you wish to advance to.

You can also insert and delete lines in the recipe. To delete the current line, select "Delete this line" from the menu. To insert a line, you must either select "Insert ingredient after this line" or "Insert fat / water / yield change after this line". If you insert an ingredient, you will be prompted to name the ingredient in the same way as in the recipe coding. If you insert a modifier line (fat / water / yield change), you will be prompted in the same way as in recipe coding.

To change just a part of the line, select "Change this line" from the Recipe Line Editor Menu. Depending on which type of line you are changing, a different menu will appear. If you are changing the recipe header, figure 3.11 will display.

Figure 3.11 Edit Recipe Header Menu


```
RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

File:  JUNK.REC          Recipe:  RECCODE001                   Line: 00
Messages:

Recipe name:    RECIPE

Recipe code:    RECCODE001

Food Group:     111      <-- Change? N



        Press RETURN or arrows to move field, DO or Enter to select.
     Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.

               Current name of recipe in file.  Type a new one.
```

You can then change the recipe name or code by moving the cursor to the appropriate field (with the cursor keys or RETURN) and then typing a new name or code. To change the food group for the recipe, answer "Y" in the "Change?" field. When you complete the menu with ENTER (or DO), the Food Group Selection Menu (Figure 3.4) will appear to let you choose a new food group. When this process is completed, the Recipe Line Editor Menu will reappear.

If you are changing an ingredient line, the menu depicted in figure 3.12 will appear.

Figure 3.12 Edit Ingredient Line Menu

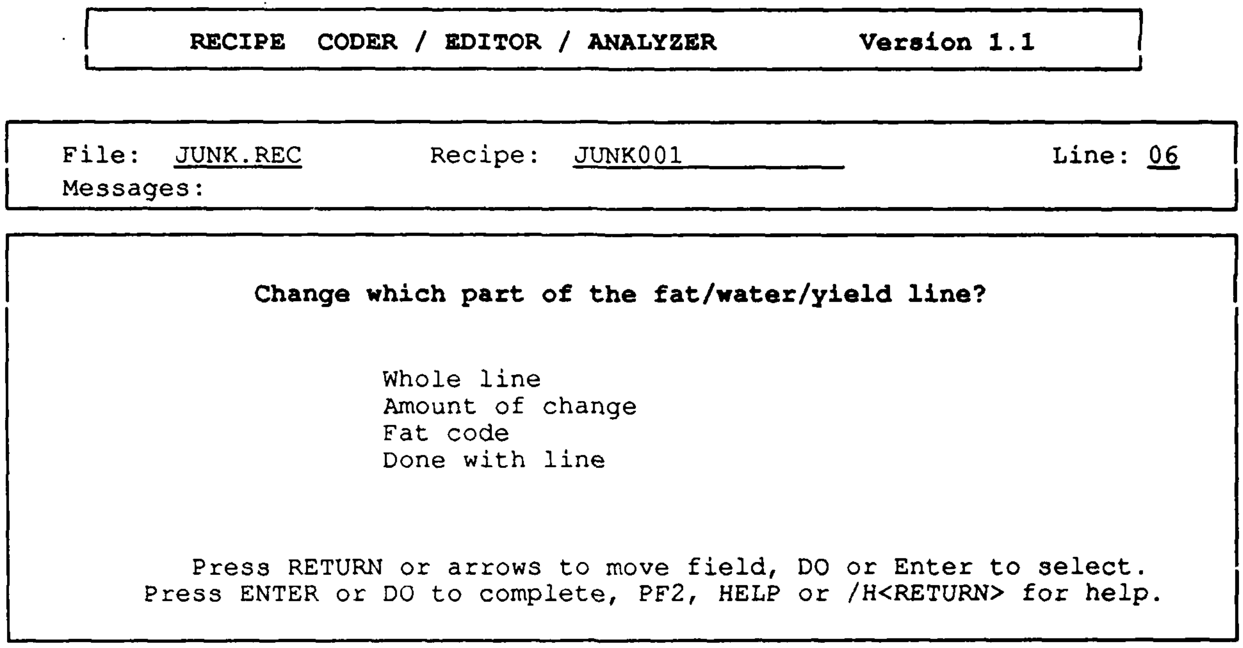
```
      RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

   File:  JUNK.REC         Recipe:  JUNK001                       Line: 02
   Messages:

                    Change which part of the ingredient line?

                              Whole ingredient
                              Unit of measure
                              Amount of ingredient
                              Process codes
                              Done with changes

             Press RETURN or arrows to move field, DO or Enter to select.
          Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.
```

If you select "Whole ingredient", you will be prompted to enter an ingredient, its units, and the amount. The other choices allow you to change only one part of the ingredient line. When all changes are complete, select "Done with changes" to return to the Recipe Line Editor Menu.

If you are changing a modifier line (fat / water / yield change), the menu that will appear after you select "Change this line" from the Recipe Line Editor Menu is shown in figure 3.13.

If you select "Whole line" from this menu, you will be prompted to enter a modifier in the same way as in the recipe coder. You can also change only the amount of the change, or the fat code (if applicable). When all changes are complete, select "Done with changes" to return to the Recipe Line Editor Menu.

Figure 3.13 Edit Ingredient Modifier Menu



```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

  File:  JUNK.REC         Recipe:  JUNK001                     Line: 06
  Messages:

              Change which part of the fat/water/yield line?

                          Whole line
                          Amount of change
                          Fat code
                          Done with line

        Press RETURN or arrows to move field, DO or Enter to select.
    Press ENTER or DO to complete, PF2, HELP or /H<RETURN> for help.
```

When all editing is complete, select "Exit editor" from the Recipe Line Editor Menu. This will return you to the Recipe Editor Menu (Figure 3.8). You can then return to the Main Menu by selecting "Exit recipe editor".

3.2.3.4 Creating Recipe Sub-set Files

Since a recipe file can contain any number of coded recipes, it is possible that you may have thousands of recipes in one file. For example, you may want to put all of the A-ration recipes into one file. In that case, you may also want to create a smaller version of the recipe file containing only the recipes you need for a particular survey. To do that, select "Create a recipe sub-set file" from the Main Menu. You will then be prompted to enter the recipes you want by the menu in figure 3.14.

First, you must type the name of the recipe file from which you are extracting recipes. Then, type the name you wish to call the new sub-set file. You can then enter individual recipe codes by just typing the code and pressing RETURN. If the recipe is found, it will be added to the sub-set file. If not, an error message will be displayed. You can continue adding recipes in this manner. When you are done, press PF4 to return to the Main Menu.

If it is known in advance exactly which files are to be extracted into the sub-set file, you can prepare a batch file ahead of time before running the Recipe CEA program. This batch file can be created using the standard VAX editor and must be named with the .BAT extension. Each line in the batch file contains a recipe code number to be extracted. There may also be comment lines beginning with an asterisk (*) in column one. Here is an example of what a batch file may look like:

```
Column 1
       |
       * This file contains recipe codes for extracting from a
       * recipe file to create a sub-set file:
       JUNK001
       JUNK002
       JUNK003
```

When this file is used in the recipe sub-set creation, the three recipes JUNK001, JUNK002, and JUNK003 will be extracted from the recipe file and put into the new recipe sub-set file. The sub-set file can then be edited, appended, or anything else you can do with a recipe file.

Figure 3.14 Create Recipe Sub-set Menu

```
     RECIPE CODER / EDITOR / ANALYZER          Version 1.1

    Type file name, RETURN, or PF4 to cancel:       junk.rec
    Type name for output subset file <PF4 to cancel>:   junk1.rec

  Type code for recipe to extract, or name of
       batch file containing recipe codes <ends
       with .BAT> or PF4 to complete:
```

3.2.3.5 Combining Recipe Files

Since it is possible to create recipe sub-set files, it may also sometimes be necessary to merge several smaller recipe files into one larger file. This can be done by selecting "Combine coded recipe files" from the Main Menu. You will then be prompted by figure 3.15.

Figure 3.15 Combine Recipes Menu

```
RECIPE CODER / EDITOR / ANALYZER        Version 1.1

    Type output file name or PF4 to cancel:    junkl.rec

Type name of coded recipe file to combine with
  output file, or batch file containing file
  names <ends with .BAT> or PF4 to complete:
```

First, you are prompted to name the output file. This file will then contain the contents of all the files you are combining. You can then enter individual file names by typing the name and pressing RETURN. If the file is found, it will be added to the output file. If not, an error message will be displayed. You can continue adding recipe files in this manner. When you are done, press PF4 to return to the Main Menu.

If it is known in advance exactly which files are to be combined, you can use a batch file in the same manner as in the previous section. Prior to combining recipe files, edit the recipe names and codes so that they are unique.

3.2.3.6 Analyzing Recipe Files

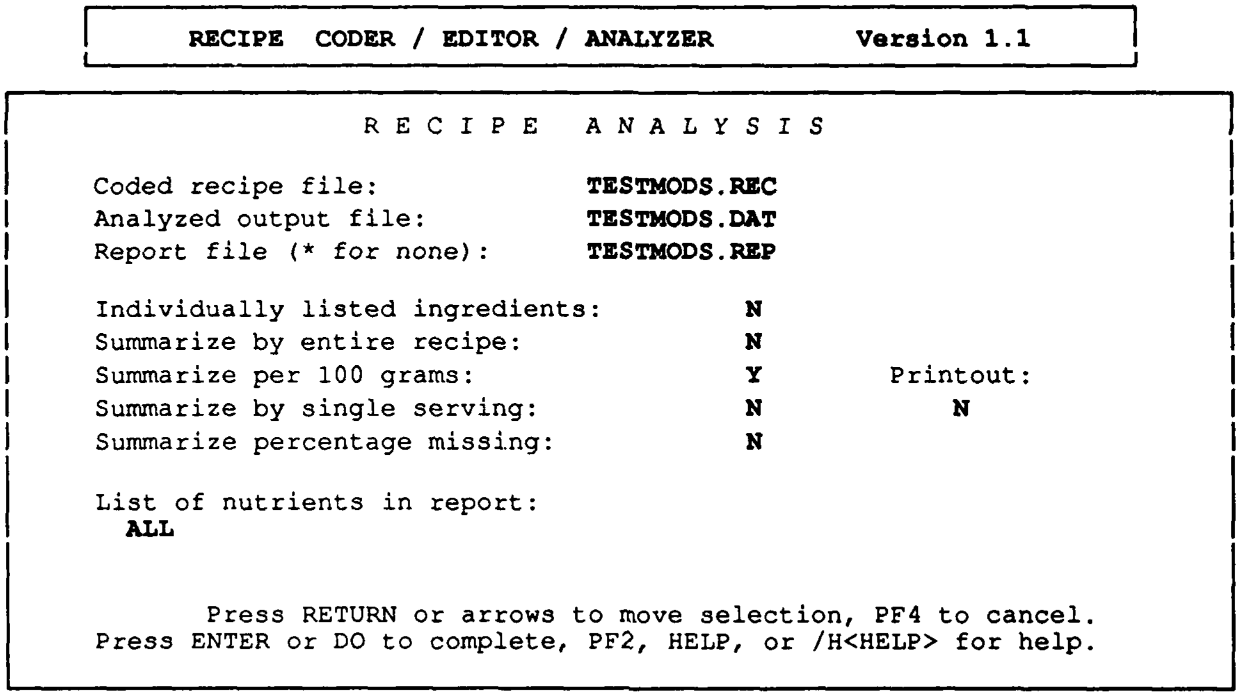
Once a coded recipe file has been created, you may wish to analyze it for nutritional composition. From the Main Menu, select "Analyze a coded recipe file" to bring up the recipe analysis menu as depicted in figure 3.16.

In the first field, type the name of the coded recipe file to be analyzed. In the second field, type the name you want to give to the analyzed output file. This file will be used later in the statistical analysis.

The third field contains the name of the report file. This file is a readable file which you can print out and examine. The report will contain the nutritional information for all of the recipes in the coded file summarized by the options selected below the file names. For each of the options, you can enter 'Y' or 'N' (for yes or no). If 'Y' is selected for "Printout", the report file will be automatically printed when complete on the default printer.

If the first option, "Individually listed ingredients", is selected, the nutrients for every ingredient in the recipe will be included. If "Summarize by entire recipe" is selected, the nutrients for all of the ingredients in the entire recipe will be totaled. "Summarize by 100 grams" will produce a summary of the nutrients based on a 100 gram serving. "Summarize by single serving" will produce a summary based on the

Figure 3.16 Recipe Analysis Menu



```
        RECIPE  CODER / EDITOR / ANALYZER          Version 1.1

                         R E C I P E   A N A L Y S I S

     Coded recipe file:               TESTMODS.REC
     Analyzed output file:            TESTMODS.DAT
     Report file (* for none):        TESTMODS.REP

     Individually listed ingredients:          N
     Summarize by entire recipe:               N
     Summarize per 100 grams:                  Y          Printout:
     Summarize by single serving:              N              N
     Summarize percentage missing:             N

     List of nutrients in report:
       ALL


            Press RETURN or arrows to move selection, PF4 to cancel.
     Press ENTER or DO to complete, PF2, HELP, or /H<HELP> for help.
```

actual coded serving size. If a serving size is not coded, the program will default to 100 servings per recipe. And finally, "Summarize by percentage missing" will produce a summary showing what percentage of the ingredients were missing data on the reported nutrients. For example, if 2 out of 10 of the ingredients in a particular recipe were missing data on cholesterol, then the summary would indicate that 20% of the cholesterol data in the recipe were missing. In the present database, most nutrients are either listed for all ingredients, or none. So, the summary will tend to show a lot of 0% and 100% figures. However, it is possible that a certain nutrient is present for only some ingredients, and therefore the percentage missing would be between these extremes.

In the last field in the menu, you can list the nutrients you wish to show in the summary. This is actually a double field since the list could easily go beyond one screen line. If you wish to see all nutrients on the summary, the word "all" should be typed into the field. You can also type specific nutrient numbers. To see what numbers correspond to what nutrients, type a question mark in the field (?) and press ENTER. The help menu will be displayed in two screens, figure 3.17 and figure 3.17.1.

The user can also select the desired nutrients from the screen. For example, suppose you want your report to contain information on only water, calories, protein, and fat. These are listed as nutrients 1 through 4.

So, you could type the following into the menu field:

1-4

Suppose in addition you want to show sugar. Sugar is listed as nutrient number 7, so you could type the following:

1-4,7

A chart correlating the nutrient numbers and abbreviations with the actual nutrient names is included in Appendix A. The units of measure for the nutrients are also included.

Once you have filled in all of the menu fields that you want, you may press ENTER to start the analysis, or PF4 to cancel back to the Main Menu. Whatever you type into the Recipe Analysis Menu field will be retained for the next time. So the chances are that you will not have to retype every field the next time you run the analysis.

Figure 3.17 Nutrient Help Menu

```
      RECIPE  CODER / EDITOR / ANALYZER           Version 1.1

Type one or a combination of the following separated by commas:

ALL:  Numbers for all 109 nutrients.  This is used alone.
LOW RANGE_HIGH RANGE:  This gives all the numbers from the low to the high.
NUMBER: This will give you only this nutrient.

    1.   H2O        2.   CAL        3.   PROT       4.   FAT
    5.   CHO        6.   SUC        7.   SUG        8.   COM
    9.   CAF       10.  DFIB       11.   DFI       12.   DFS
   13.   ASH       14.     C       15.   THI       16.   RIB
   17.   NIA       18.    B6       19.   FOL       20.   B12
   21.   BIO       22.   PAN       23.   ARE       24.   AIU
   25. CARRE       26.   CAR       27.     D       28.  ETOT
   29.  TTOC       30.   TOC       31.  VITK       32.    CA
   33.     P       34.    MG       35.    FE       36.   FEH
   37.   FEN       38.    ZN       39.     I       40.    CU
   41.    MN       42.    FL       43.    CR       44.    SE
   45.    MO       46.     K       47.    NA       48.    CL
   49.     S       50.    AL       51.    BA       52.    BO

    Press <CR> to continue:
```

Figure 3.17.1 Nutrient Help Menu

```
      RECIPE  CODER / EDITOR / ANALYZER          Version 1.1
 45.     MO           46.      K           47.     NA           48.     CL
 49.      S           50.     AL           51.     BA           52.     BO


  Press <CR> to continue:
 53.     SR           54.     SI           55.     VA           56.     CO
 57.     NI           58.    ARS           59.    TIN           60.    SAT
 61.     S4           62.     S6           63.     S8           64.    S10
 65.    S12           66.    S14           67.    S16           68.    S18
 69.    MON           70.    M16           71.    M18           72.    M20
 73.    M22           74.    POL           75.   P182           76.   P183
 77.   P184           78.   P204           79.   P205           80.   P225
 81.   P226           82.   CHOL           83.    PHY           84.    HIS
 85.    ILE           86.    LEU           87.    LYS           88.    MET
 89.    PHE           90.    THR           91.    TRY           92.    VAL
 93.    ALA           94.    ARG           95.    ASP           96.    CYS
 97.    GLU           98.    GLY           99.    PRO          100.    SER
101.    TYR          102.     MC          103.     TP          104.    CAF
105.    ETH          106.   CARN          107.    INO          108.    PHT
109.    CHL

 Press <CR> to continue:
```

After you press ENTER to start the analysis, a status box will appear at the bottom of the menu. This will show you which recipe is currently being analyzed. The status box is seen in figure 3.18.

The word "Analyzing" will be blinking and the recipe names will be updated as the analysis progresses.

When the analysis is complete, a message will indicate that it is complete and that the report is being printed (if selected). Then, the program will automatically return to the Main Menu.

If you wish to analyze only a few recipes in a coded file, you must first create a subset file as described previously (see section 3.2.3.4) and then analyze that subset.

Figure 3.18 Recipe Analysis Menu: Analyzing

```
RECIPE  CODER / EDITOR / ANALYZER          Version 1.1
```

```
                    R E C I P E   A N A L Y S I S

   Coded recipe file:              TESTMODS.REC
   Analyzed output file:           TESTMODS.DAT
   Report file (* for none):       TESTMODS.REP

   Individually listed ingredients:       N
   Summarize by entire recipe:            N
   Summarize per 100 grams:               Y         Printout:
   Summarize by single serving:           N            N
   Summarize percentage missing:          N

   List of nutrients in report:
     ALL

 ANALYZING:    RECIPE 1:   BASE FORM WITH NO FAT/WATER CHANGES
```

3.3 *Recipe Coder / Editor / Analyzer* Programmer's Guide

3.3.1 Introduction

The Recipe Coder / Editor / Analyzer (CEA) is an interactive program on the VAX for creating and editing coded recipe files and then analyzing them for nutritional composition. This document describes some of the details on how the program works. It is intended more for the programmer, not the user. For details on how to use the program, see the User's Guide.

3.3.2 File Formats

The Recipe CEA uses several input files for the ingredients database, units of measure, and retention factors. It also creates and reads coded recipe files. All of these files are Fortran keyed indexed files. In a keyed indexed file, all records are a fixed length and a certain field is designated as the "key". The key field is just a certain range of character positions. For example, if a keyed indexed file contains records of length 80, we might specify that characters 1 through 10 are to be the key. Each record must have a unique key, and Fortran automatically maintains the file in order by sorting on the key.

To open an existing keyed indexed file, the following Fortran OPEN statement would be used. Note that the words in all capitals are typed exactly as shown, whereas the words in lowercase italics are variable:

```
OPEN (UNIT=lu, FILE="filename", STATUS="OLD",
          ACCESS="KEYED",ORGANIZATION="INDEXED",
          KEY=(start:end:datatype),RECL=recordlength,
          ERR=linenum)
```

The unit number, specified by lu, can be whatever logical unit the program is using for the file. The start and end numbers specify the starting and ending character positions in each record to be used as the key field, and the datatype is the type of information in the field such as INTEGER, CHARACTER, REAL, etc. In the Recipe CEA, all keyed indexed files have CHARACTER keys. The recordlength is the number of bytes (or characters) in each record, and the linenum is the program label number to transfer to on error conditions. At that point, the program can display the appropriate error messages and take whatever action is necessary. To create a new keyed indexed file, the status would be changed from "OLD" to "NEW".

The ingredients file contains the entire nutrients database. The file is prepared in advance by running the Dumptable program. Dumptable translates the Oracle table into the Fortran format. It was discovered in an earlier version of the Recipe Coder that reading Oracle tables directly in a Fortran program takes too much time. So for that reason, the Fortran file is created in advance. Once it is read into memory, accessing it is very fast. Note that Dumptable only needs to be run when a major update is performed on the Oracle table. The ingredients file is opened to logical unit 15 with the following OPEN statement:

```
OPEN (UNIT=15, FILE="INGREDV6.DAT", STATUS="OLD",
          ACCESS="KEYED", ORGANIZATION="INDEXED",
          KEY=(1:10:CHARACTER), RECL=165, ERR=990)
```

Figure 3.19 shows the record format for the ingredients file. There are two important things to notice. First, the ingredient name field used to be 60 characters. It was changed to 80 when it was discovered that many of the names were identical up to the 60th character. The second thing to notice is that the sum of the number of bytes in one record exceeds the number declared in the file OPEN statement, i.e. 165. This is okay as long as every OPEN statement, including the one in the Dumptable program, is consistent. Fortran will just allocate whatever extra amount is needed. So, for historical reasons, 165 is retained to avoid incompatibilities.

Figure 3.20 shows the record format of the unit of measure file. This file is also prepared in advance by running Dumptable. It is opened to logical unit 1 in exactly the same way as the ingredients file except that the record length is 420. Again, this record length is not actually long enough, but for backward compatibility, it is retained.

Figure 3.21 shows the group retentions file which is also prepared in advance with Dumptable. This file is used to correlate processing codes (or retention factors) with certain food groups. For example, we don't want any of the foods in the "beer" group to have processing codes like "saute". Notice that the food group is shown as a three character code instead of three real numbers as in the ingredients file. Since all three numbers are always between 0 and 9, the three numbers can be concatenated into a three character string. Like the other two files, the group retentions file is opened with a record length of 1080 even though the total number of bytes per record is more. But once again, this is retained for backward compatibility.

Figure 3.19 Ingredients File Record Format

BYTES		DATA TYPE
10	Ingredient Code (KEY)	Character
80	Ingredient Name	Character
4	Major Food Group	Real
4	Minor Food Group	Real
4	Sub-minor Food Group	Real
20	(Not currently used)	
109x4	109 Nutrients	Real

Figure 3.20 Units Of Measure File Record Format

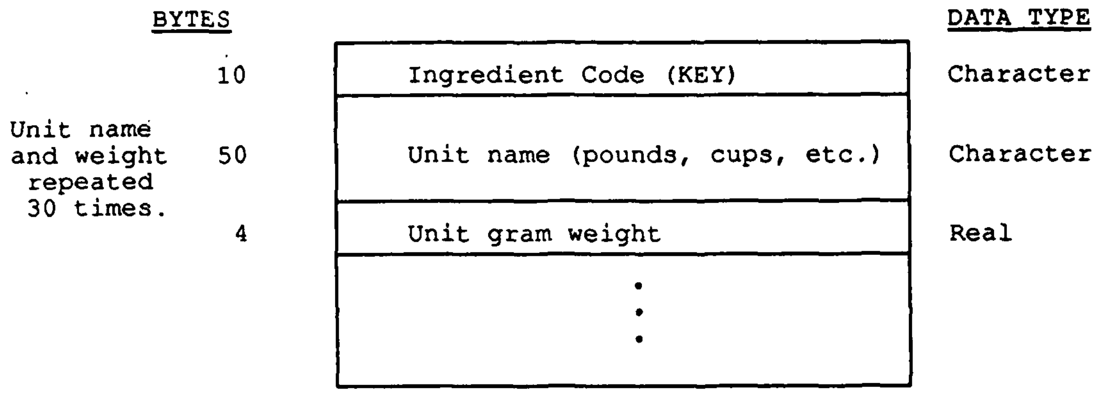


Figure 3.21 Group Retentions File Record Format

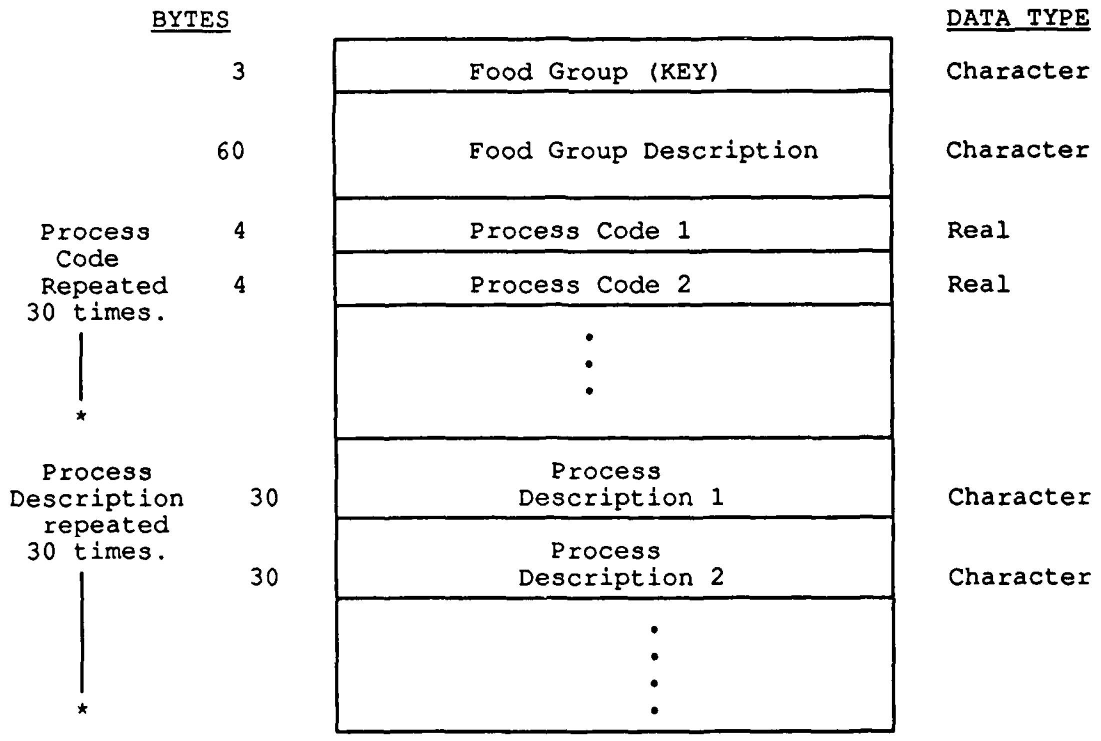


Figure 3.22 shows the retentions file which is prepared in advance with Dumptable. This file correlates the retention codes (or process codes) with the actual retention factors for all 109 nutrients. Again, like the other files, the total record length

exceeds the declared record length of 130. Notice also that the process code is a 4 character string instead of a real number as in the group retentions file. This can be done since none of the process codes has more than 4 digits. Also notice that the process description is longer than in the group retentions file. The descriptions in the group retentions file have been truncated. Actually, it is not really necessary to keep the descriptions in the group retentions file since they can be found by looking in the retentions file, but for backward compatibility, they are maintained.

Figure 3.22 Retention File Record Format

BYTES		DATA TYPE
4	Process Code (KEY)	Character
60	Process Description	Character
109x4	109 Retention Factors	Real

In addition to the 4 database files, the Recipe CEA also creates keyed indexed files for the coded recipes. These files are used internally by the program. The user would normally translate the file into a readable format before examining the coded recipes. Figure 3.23 shows the format of the coded recipe files. Notice that in this case, the number of bytes in each record is exactly the same as the number declared in the OPEN statement, i.e. 126.

There are also a couple of other things that should be mentioned about the way this file is used. First, the recipe code and the line number within the recipe are combined to form the key. The recipe code cannot be used alone because each record in a keyed indexed file must have a unique key. By concatenating the line number, we are assured of every record having a unique key.

The other thing to point out about this file is that, except for the first two fields, the remainder of the record can be interpreted two ways. The left column shows how the first record of each recipe is interpreted. The right column shows how the other lines of the recipe are interpreted. Rather than showing the data types, each field shows the equivalent Fortran format in parentheses.

There are also two other types of files: Resource files and Help file. Resource files contain the templates for menus. There is one resource file for each menu. For more detail on the format of resource files, see the Menu Manager document. Help files are just regular ASCII text files. There is one help file for each resource file. Help can be requested by the user at any menu. When the user types PF2 (or HELP on some terminals), the help file is read in and displayed in screen sized chunks.

Figure 3.23 Coded Recipe Files Record Format

			Bytes
KEY	Recipe Code (A10)		10
	1 Space, then Recipe Line Number (1X,I2.2)		3
	1 Space, then Recipe Name (1X,A60)	Line Type (A2)	2
		1 Space, then Ingredient Code (1X, A10)	11
		1 Space, then Ingredient Amount (1X, F9.2)	10
		1 Space, then Unit Label (1X, A50)	51
		1 Space, then Gram Weight (1X, F10.1)	11
		1 Space, then 3 Process Codes (1X, 3A4)	13
	1 Space, then Food Group (1X,A3)	Not Used	15

One other file is the Food Group file. This file is read and parsed when the user is selecting a food group. The file is just a regular ASCII text file, but there are certain keywords and symbols embedded for easier parsing. The format of the file is is shown in figure 3.24.

Figure 3.24 Food Group File Format

```
Column one
    |
    |
    Comments at the beginning of the file.
    +MAJOR
    #0
    1  Dairy
    2  Meats
    3  Eggs
          ...
    #END
    +MINOR
    #1  (Dairy)
    11  Fluid Milk
    12  Cream and cream substitutes
          ...
    #2  (Meats)
    21  Beef
    22  Pork
          ...
    #3  (Eggs)
          ...
    #END
    +SUBMINOR
    #11  (Fluid milk)
    111  Milk, fluid
    112  Milk, concentrated
          ...
    #12  (Cream, etc.)
    121  Dairy cream
          ...
    #21  (Beef)
          ...
    #END
```

3.3.3 Overall Program Structure

Figure 3.25 shows the overall structure of the Recipe CEA. After starting the program, initialization is performed. This initialization can take 3 or 4 minutes because it reads the entire ingredients database into memory. The ingredients database is a Fortran keyed indexed file which is prepared in advance with the Dumptable program. See the Database Translation documentation for more detail.

After the initialization, the Main Menu appears. The user can then choose one of the items to perform that function. The structure of the code that handles the Main Menu (and any other menu) is covered in more detail in the Menu Manager document. Depending on what the user chooses, the main routine calls the appropriate subroutine. That subroutine may in turn put up another menu. In that way, menus can easily be nested. When the function is completed, control is returned to the main routine and the Main Menu is redisplayed.

Whether the user chooses "Create new recipe" or "Append to existing recipe" from the Main Menu, the same subroutine is called. A parameter is passed to that subroutine to indicate whether the coded recipe file should be opened as 'NEW' or 'OLD'. Other than that, the rest of the recipe coding is the same for both choices.

While the user is coding, the recipe is being composed in a temporary scratch file. This allows for the possibility of deleting recipe lines before committing them to the recipe. When the recipe is complete, the lines are copied to the recipe file. If the user cancels, the scratch file is closed and discarded without updating the recipe file.

If the user chooses "Check / Translate recipe file" from the Main Menu, a subroutine is called which reads the coded recipe file line by line and translates it into a more readable format. The translated file is a regular ASCII text file and it can be printed directly from the program.

Recipe editing is somewhat more complicated. When this is chosen from the Main Menu, another menu appears. The user can choose to duplicate a recipe, delete a recipe, or edit a recipe on a line by line basis. If duplication is selected, a menu appears prompting the user for a new recipe name and code. Then, the entire recipe is read into an array (one line per array element), the new name and code are substituted into the array, and then the whole recipe is written out to the recipe file. Since the recipe file is a keyed indexed file, and the duplicate recipe has a different recipe code, it does not wipe out the original version.

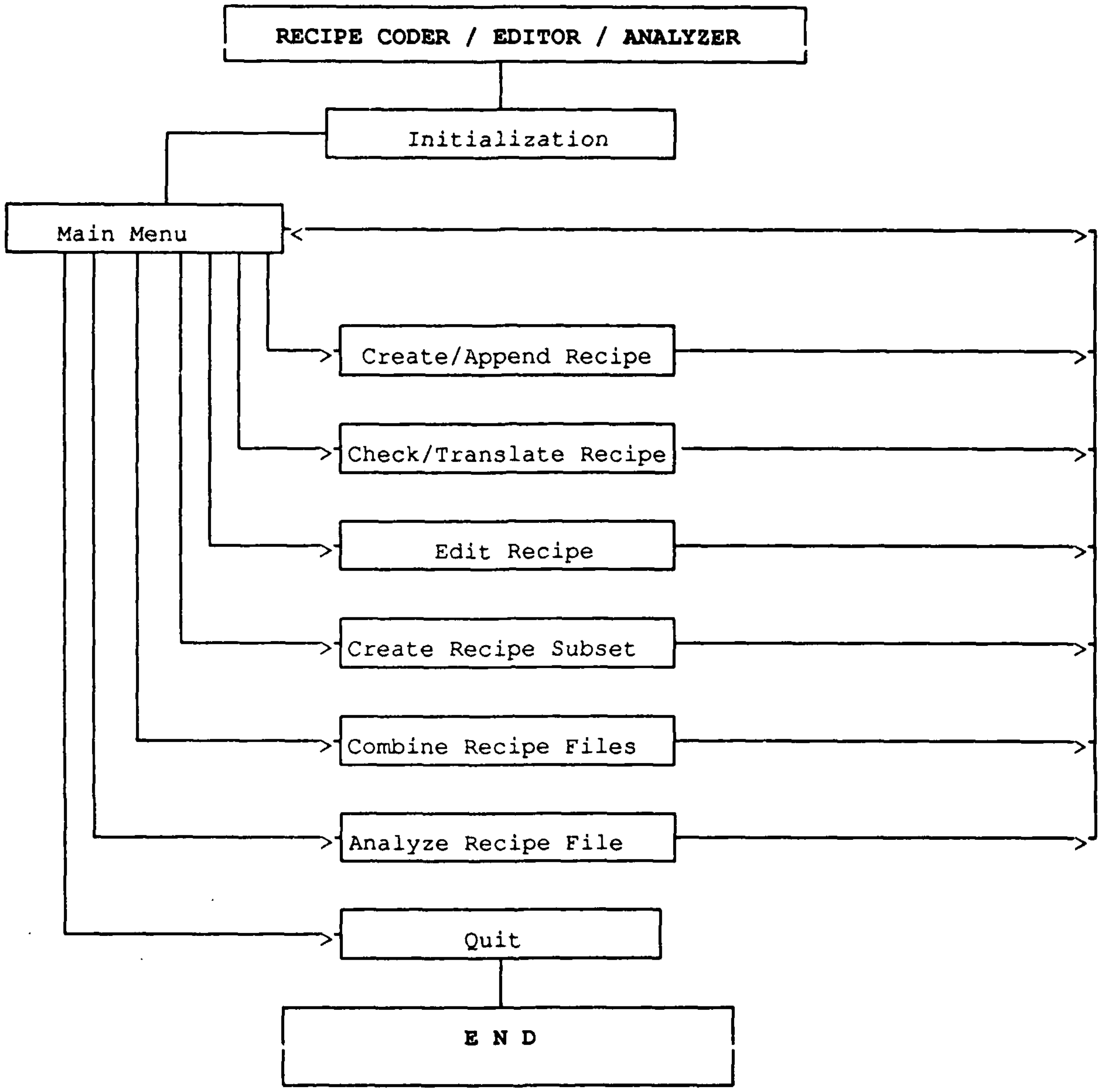
Deleting a recipe from a coded file is fairly simple. All of the lines for a recipe are located by reading the file with the appropriate key (the recipe code), and then deleted with the Fortran DELETE statement.

When the user chooses to edit the recipe on a line by line basis, the entire recipe is read into an array in the same way as with duplication, but it is also deleted from the coded recipe file. This is necessary in order to enable the file to be updated later. The actual editing is performed on the array, not the file. When the editing session is complete, the array is written out to the recipe file.

To create a recipe subset file, the user is prompted for the recipe codes to extract from the current file, or the name of a batch file containing the recipe codes. If a batch file is being used, each name is parsed out of the file and processed one by one as if the user had typed them. Each recipe is then read into an array as above and then written out to the subset file.

Combining recipe files is similar to creating subset files. The user is prompted for each of the recipe files, or a batch file containing the names of the recipe files. Then the files are read in line by line and written out to the combined file.

Figure 3.25 Recipe Program Structure



3.3.4 Recipe Analysis

The recipe analysis of a coded recipe file is fairly simple from the user point of view, but it requires some explanation from the programming point of view. When the user chooses "Analyze recipe file" from the Main Menu, the Recipe Analysis Menu appears (Figure 3.16). In this menu, the user types the name of the file to be analyzed, the output files, and various analysis options. Then, the user types ENTER, and the analysis proceeds automatically.

The program reads in each line from the coded recipe file one by one. If the line is the first line of a recipe, any pending recipe analysis is completed, and the results are written out to the report file. Then the next recipe is analyzed.

One array is used to keep track of the total recipe weights for each of the possible 9 sub-recipes, and another large array keeps track of the totals of all 109 nutrients for each of the possible sub-recipes. As the file is read line by line, the ingredients are found in the database and the total weights and nutrients are accumulated for all the ingredients in the current sub-recipe. When processing codes for an ingredient are encountered, the weights and nutrients for that ingredient are adjusted according to the retentions file.

When fat, water, or yield changes are encountered, they are saved temporarily in an array to be processed when the sub-recipe is completed. This enables the user to code the recipe with yield changes anywhere in the sub-recipe as opposed to strictly at the end. When the sub-recipe is completed, the yield changes are processed for that sub-recipe and the weights and nutrients are adjusted accordingly. Adjustments to the final fat or water percentage are only processed for the total recipe, not for individual sub-recipes.

Water gains or losses, when expressed as a percentage of the total recipe weight or total water weight, are very straightforward. The amount of water coded as a gain or loss is simply added to or subtracted from the total weight and nutrients for the current sub-recipe. Fat losses expressed as a percentage of the total recipe or total fat are also straightforward. The amount of fat coded as a loss is simply subtracted from the total weight and nutrients for the current sub-recipe. However, this also involves adjusting the fatty acids by the amount proportional to the fat loss.

Fat gains are handled somewhat differently. If a certain percentage of the total weight or fat weight is coded as a fat gain, the recipe analysis must look for the source of fat in the ingredients database and adjust all of the nutrients accordingly. For example, if the recipe is coded to have a fat gain of 10% of the original fat, the source of fat must also be in the coded recipe file. If the source of fat is butter, the analysis program must find out how much butter needs to be added in order to gain 10% of the original fat. So, if the original fat weight was 100 grams, we would need to add 10 grams of butter in order to increase the fat by 10%. This analysis is more complicated because butter contains more nutrients than just fat. So, the ingredients database is consulted for butter and the appropriate amounts of all nutrients plus fat and water are adjusted to account for the gain of 10% fat.

Adjusting the final percentage of fat and water is even more complicated. In the simplest case, only the final percentage of water is coded. This is different from the above cases because we are coding the final percentage of the total recipe weight that is water instead of the actual percentage of water that is gained or lost. In other words, instead of saying that some percentage of the water is lost or gained, we are saying that the final total recipe weight consists of a certain percentage of water. At first, it is not obvious whether this is a gain or a loss. To compute the actual amount of water gained or lost, the following formula is used:

$$Y = \frac{(W * H) - T1}{1 - H}$$

where Y is the resulting water adjustment in grams,
W is the total recipe weight before adjustment,
H is the final percentage of water desired,
T1 is the total water weight before adjustment.

All percentages in these formulas are actually expressed as ratios, so 25% would be expressed as 0.25 in the formula. The resulting adjustment, Y, is added to the total recipe weight and the total water weight in the accumulated nutrients (since water is one of the nutrients) to yield the final results. If there is a water loss, Y will be negative; for a gain, Y will be positive.

The next most complicated case occurs when the final fat percentage is specified. In this case, we must first figure out if the final percentage represents a gain or a loss. If it is a loss, we just want to adjust the recipe weight, fat weight, and fatty acid weights. But if it is a gain, we have to adjust for the addition of some fat source. To determine whether the final fat percentage is a gain or a loss, the following formula is used:

$$FO = FG / W$$

where FO is the calculated original percentage of fat,
FG is the original amount of fat in grams,
W is the total recipe weight before adjustment.

The original percentage of fat, FO, is then compared to the coded final percentage to determine if there is a fat loss or gain. If there is a loss, the amount of fat lost is calculated as follows:

$$X = W - \left(\frac{W - T4}{1 - F} \right)$$

where X is the resulting fat loss in grams,
W is the total recipe weight before adjustment,
T4 is the total fat weight before adjustment,
F is the final percentage of fat desired.

The resulting fat loss, X, is then subtracted from the total recipe weight and the total fat weight. The fatty acids are decreased proportionately.

If FO is compared against the coded final percentage of fat and a fat gain is determined, then the appropriate amount of fat source must be added. The amount of the fat source ingredient which needs to be added is calculated as follows:

$$XS = \frac{T4 - (W * F)}{F - C4}$$

where	XS is the amount of the fat source to be added,
	T4 is the total fat weight before adjustment,
	W is the total recipe weight before adjustment,
	F is the final percentage of fat desired,
	C4 is the concentration of fat in the fat source.

The concentration of fat in the fat source, C4, is the percentage (or ratio) of actual fat in the ingredient as compared to the total weight of the ingredient. For example, if 10 grams of butter contains 7 grams of fat, then the concentration factor, C4, would be 0.7. This factor is determined by searching in the ingredients database for the fat source ingredient and finding the amount of fat per 100 grams of ingredient.

Notice that in order for this formula to make sense, the concentration factor must be greater than the final percentage. If they are equal, the formula will fail when it attempts to divide by zero. If the concentration is less than the final percentage, the amount of fat source to be added, XS, will be negative. This does not make physical sense because you can't remove butter and expect the fat percentage to increase. So, it is possible for the user to code a final fat percentage that does not make sense. Unfortunately, this cannot be detected at coding time. It can only be detected during the analysis in which case the user would have to re-edit the recipe and then re-analyze it.

The most complicated case for yield adjustment occurs when both a final fat percentage and a final water percentage are specified. If the final fat percentage represents a gain, then the addition of a fat source will also result in a reduction of water. So, the wo factors must be calculated simultaneously.

First it must be determined if the final fat percentage represents a gain or a loss. To do that, the following formulas are used:

$$Z = \frac{W - T1 - T4}{1 - (F + H)}$$

$FG = F * Z$
$WG = H * Z$

where Z is the final adjusted total recipe weight,
W is the total recipe weight before adjustment,
T1 is the total water weight before adjustment,
T4 is the total fat weight before adjustment,
F is the final percentage of fat desired,
H is the final percentage of water desired,
FG is the final amount of fat in grams,
WG is the final amount of water in grams.

If the final amount of fat in grams, FG, is less than the original amount of fat in grams, T4, then we have a fat loss. On the other hand, if FG is greater than T4, we have a fat gain and we must account for the gain in fat source which gives us the correct gain in actual fat.

If there is a fat loss, we can find the amount of fat lost and the water gained or lost with the following formulas:

$X = T4 - FG$

$Y = WG - T1$

where X is the resulting fat loss in grams,
Y is the resulting water adjustment in grams,
FG is the final amount of fat in grams,
WG is the final amount of water in grams.
T1 is the total water weight before adjustment,
T4 is the total fat weight before adjustment,

The resulting fat loss, X, is then subtracted from the total recipe weight, and the total fat weight. The fatty acids are adjusted accordingly. Then, the resulting water adjustment is added to the total recipe weight, and the total water weight. If there is a water loss, Y will be negative; for a gain, Y will be positive.

If after calculating the final weight of fat in grams, FG, we discover that there is a gain in fat, we must calculate the amount of fat and water adjustments simultaneously using the following formulas:

$$X = \frac{T4(H - 1) + F(W - T1)}{F(C1 - 1) - C4(H - 1)}$$

$$Y = \frac{T4 + (X * C4)}{F} - X - W$$

where X is the resulting fat source adjustment in grams,
Y is the resulting water adjustment in grams,
T1 is the total water weight before adjustment,
T4 is the total fat weight before adjustment,
C1 is the concentration of water in the fat source,
C4 is the concentration of fat in the fat source,
F is the final percentage of fat desired,
H is the final percentage of water desired,
W is the total recipe weight before adjustment.

In these formulas, X represents the amount of the fat source in grams, not the amount of fat in the recipe. In other words, X is the amount of butter to add, not the amount of fat. The butter in turn contains water which affects that computation. So, to get the final results of the fat and water adjustments, we need to add X grams of fat source as an ingredient, and add (or subtract) Y grams of water to the final weight and water weight. Since the X grams of fat are being added as an ingredient, all of the other nutrients, including water, will also be adjusted.

3.4 Oracle Table Translator (Dumptable)

3.4.1 Introduction

The CAN System uses Orac'e on the VAX to maintain the various databases used in the Recipe Coder / Editor / Analyzer, and possibly other programs. Oracle is used since it is good for maintaining and modifying large databases. However, using a FORTRAN program to read and sort through data in Oracle can be very time consuming. So, the program DUMPTABLE was created to translate the Oracle tables into Fortran keyed indexed files. These files can be read by the FORTRAN programs more efficiently. As more additions and modifications are made to the Oracle tables, DUMPTABLE must be run to update the FORTRAN files accordingly.

Running DUMPTABLE is fairly simple, so this document will serve as the user's guide, programmer's guide, and the maintenance manual all in one.

3.4.2 Running the Program

To start DUMPTABLE, type "RUN DUMPTABLE" from the DCL prompt. When the program starts, a header with the version date will be displayed followed by the main menu. The main menu is shown in figure 3.26.

Figure 3.26 Oracle Table Translator Main Menu

```
            ORACLE  TABLE  TRANSLATOR              1/17/89

What process would you like to run?

1.  Dump a nutrient table
2.  Dump a retention table
3.  Dump a unit table
4.  Dump a group retention table
    <CR> to exit program

      Your choice:
```

After you choose the type of table you wish to translate to Fortran format, the program will ask you to type the name of the Oracle table and the name for the output Fortran file. Most of the Oracle tables are very large and they can take anywhere from 15 minutes to several hours to translate. During that time, messages are displayed at the bottom of the screen to let you know how far the translation has progressed. When it is complete, the program returns to the main menu. Press RETURN to exit from the program.

3.4.3 Oracle Table Formats

There are currently four Oracle tables that can be translated into Fortran keyed indexed files. The current version of the ingredient's nutrient table is called "STDREFV6". It contains the ingredient codes, names, and nutrients. The Oracle table format is as follows:

FIELD NAME	FIELD TYPE	DESCRIPTION
CODE	CHAR(10)	Ingredient code.
NAME	CHAR(80)	Ingredient name.
DESC1	CHAR(20)	Reserved for future info.
DESC2	CHAR(20)	Reserved for future info.
DESC3	CHAR(20)	Reserved for future info.
DESC4	CHAR(20)	Reserved for future info.
DESC5	CHAR(20)	Reserved for future info.

DESC6	CHAR(20)	Reserved for future info.
MAJ	NUMBER(1)	Major food group.
MIN	NUMBER(1)	Minor food group.
SUB	NUMBER(1)	Sub-minor food group.
A	NUMBER(1)	
B	NUMBER(1)	
T	NUMBER(1)	
MRE	NUMBER(1)	MRE flag (1=MRE, 0=not).
ING	NUMBER(1)	
H2O	NUMBER	The remaining fields contain
CAL	NUMBER	the nutrients.
PROT	NUMBER	
...	...	
CHL	NUMBER	

For a complete list of all the nutrients contained in the database, see the users's guide Appendix A. Or, you can get the information directly from Oracle by logging into Oracle and typing "DESCRIBE STDREFV6".

The current version of the retentions table is called "RETENTIONS". It contains the process codes, names, and nutrient retention factors. For example, a certain process code may correspond to boiling green leafy vegetables. During that process, certain nutrients are lost (or gained). The Oracle table format is as follows:

FIELD NAME	FIELD TYPE	DESCRIPTION
PRO	CHAR(4)	Process code.
PROCESS	CHAR(60)	Process description.
RH2O	NUMBER	The remaining fields
RCAL	NUMBER	contain the
RPROT	NUMBER	retention factors.
...	...	
RCHL	NUMBER	

The current version of the units table is called "ALLUNITS". It contains the ingredient code, unit name, gram weight, and information source. For example, a certain ingredient may be measured in cups, tablespoons and teaspoons. For each of those units, there will be a record in the Oracle table. If an ingredient is not listed in this table, it will default to pounds, ounces, and grams. The Oracle table format is as follows:

FIELD NAME	FIELD TYPE	DESCRIPTION
CODE	CHAR(10)	Ingredient code.
LABEL	CHAR(50)	Unit label.

EPWT	NUMBER	Gram weight.
SOURCE	CHAR(3)	Source code.

The current version of the group retentions table is called "GROUPRET". It contains the retention factor codes that correspond to certain food groups. For example, a food group such as "Fluid Milk" may have certain processing codes that are applicable such as "simmer" or "boil". But other codes such as "fry" or "bake" do not apply. This table contains enough room for each food group to have up to 30 applicable processing codes. It also contains the names of the processes, although they are truncated to 30 characters. The complete 60 character names can be obtained by looking in the RETENTIONS table. The Oracle table format is as follows:

FIELD NAME	FIELD TYPE	DESCRIPTION
FDGRPC	CHAR(3)	Food group (3-digit number).
FDGRPL	CHAR(60)	Food group name.
RETC1	CHAR(4)	Retention codes corresponding
RETC2	CHAR(4)	to Process code field in
RETC3	CHAR(4)	the RETENTIONS table.
...	...	
RETC29	CHAR(4)	
RETC30	CHAR(4)	
RETL1	CHAR(30)	Truncated process descriptions.
RETL2	CHAR(30)	
RETL3	CHAR(30)	
...	...	
RETL29	CHAR(30)	
RETL30	CHAR(30)	

For a complete description of the translated Fortran files, see the programmer's documentation on the Recipe Coder / Editor / Analyzer.

3.4.4 Programming Considerations

Figure 3.27 shows the overall structure of the DUMPTABLE program. After starting the program, some basic initialization is performed. Then, the main menu appears and the user makes a selection. Depending on which table is selected, the appropriate subroutine is executed to handle translating that table to Fortran format.

Each of the tables are handled similarly except that the specific table formats are different. Figure 3.28 shows the general format for all of the table translating subroutines. The only table which is significantly different is the UNITS table. The other tables all have a one to one correspondence between Oracle records and Fortran records. But the UNITS table essentially "gathers up" all of the Oracle records that pertain to a certain ingredient and outputs just one Fortran record with all of the information. So, there will be fewer UNITS records output than are input.

Figure 3.27 Oracle Translator Program Structure

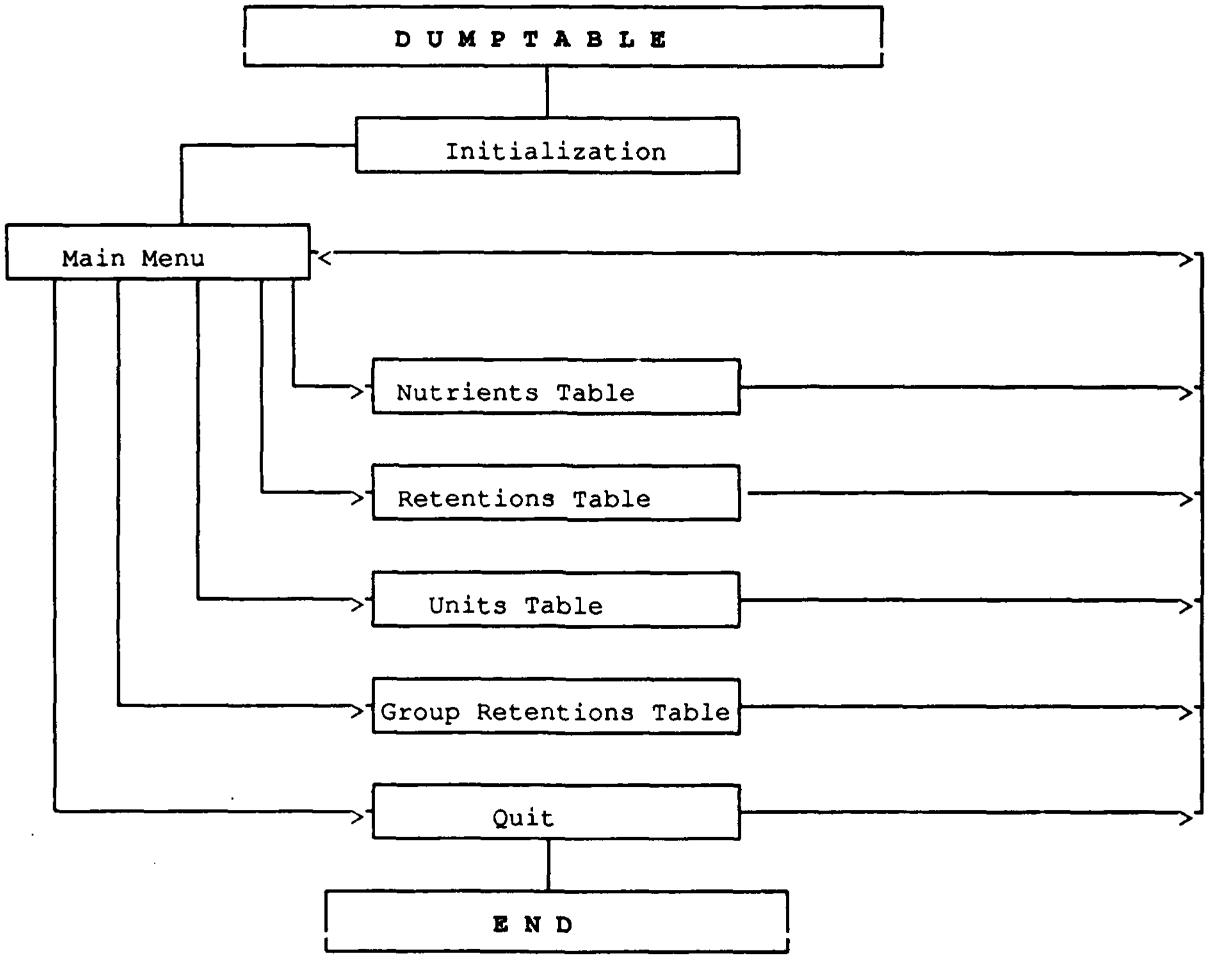


Figure 3.28 General Format Of Table Translation

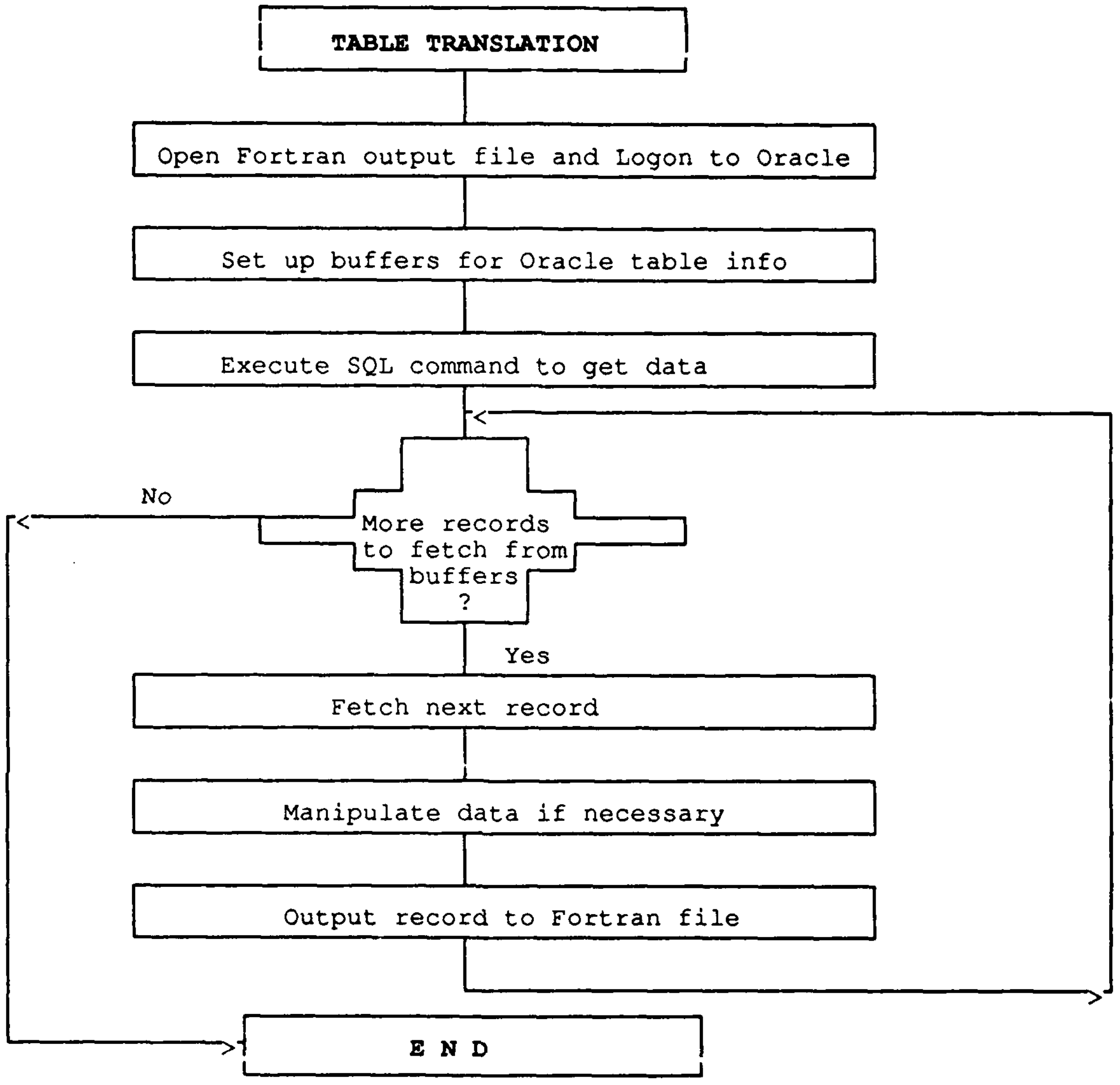


TABLE TRANSLATION
Open Fortran output file and Logon to Oracle
Set up buffers for Oracle table info
Execute SQL command to get data
More records to fetch from buffers ?
No
Yes
Fetch next record
Manipulate data if necessary
Output record to Fortran file
E N D

3.5 Menu Manager Programmer's Guide

3.5.1 Introduction

The Menu Manager is a library of subroutines which are needed for handling menus and screen forms interactively in a program. There are currently three versions of the Menu Manager: VAX Fortran, VAX C, and Force C. Since the three versions are about 95% functionally equivalent, this document will discuss how to use them in your own programs for any of the three versions. Any differences among the three versions will be noted. This document is aimed primarily at the programmer, not the user.

The menus can be used in several ways in your program. For example, a list of items can be displayed from which the user chooses one by moving the cursor through the list with the arrow keys and then pressing ENTER. Or, a menu can present the user with a blank form with fields to be filled in by the user. The fields can have default values, and numeric fields can have range checks.

This document will discuss the different ways to use menus in your program and the interfaces needed for the three versions.

3.5.2 Resource Files and Screen Forms

For every menu in your program, there is an associated "resource" file. It is possible to have more than one menu in a resource file, but that would require you to read in all of the menu resources in the beginning of the program and keep them in memory. It is easier to keep each menu in a separate resource file.

The resource file contains the template for the menu and its fields. It is created in a standard editor and it allows you to set up the screen orientation of the menu and its fields without having to do complicated subroutine calls within your program.

The resource file essentially describes the format of the menu: the locations on the screen of all the fields, the contents of the fields, the type of field, etc. The resources within a resource file are contained within braces { }. Anything outside the braces is considered to be a comment. At run time, the resource file is read in and parsed to set up the menu data structure. Here is an example of what a simple resource file might look like:

```
This is a sample resource file.  This is a comment because
it is outside the braces.

Here is the menu template which shows the locations of the
menu fields:


{ 5
                  #0

     #1                #2
     #3                #4
     #5                #6

     * 1 4 80 11
}
```

Here are the field resources which show the contents and types of fields that are in the template:

```
{ #0  PROMPT="Main Menu",BOLD }
{ #1  PROMPT="Your name:" }
{ #2  STRING="Default Name",UNDERLINE  LEN=40 }
{ #3  PROMPT="Your age:" }
{ #4  INT=21  RANGE=1,99
            ERROR="Type a number from 1 to 99!",BOLD,BLINK }
{ #5  PROMPT="Your height:" }
{ #6  FLOAT=68.0  RANGE=1.0,99.9
            ERROR="Type a number from 1.0 to 99.0!"
            NOTE="Type your height in inches." }
```

Notice that there are comments at the top of the resource file. Anything outside the braces { } is considered to be a comment. The first resource is the menu template which shows the locations of the menu fields. The first number after the open brace is a screen line number. The first line of the menu which follows the open brace will appear on the screen on that screen line number. In this case, field #0 will appear on line 5 of the terminal screen. If no line is specified, line 1 will be assumed. You can also simply space down the amount you want rather than specifying a line. For example, in this case we could have skipped 5 lines after the open brace rather than specifying line 5 after the open brace.

After that, each of the menu fields are shown in their relative positions on the screen. When the user is moving from field to field with the arrow keys, the cursor will move in ascending numeric order with the down arrow and descending numeric order with the up arrow. You can arrange the numbers in any order you want, and it is also possible to use the arrow keys differently, but more on that later.

Note: The fields are numbered starting with #1 in the VAX Fortran version. The two C versions start with field #0.

After all of the field positions are specified, there is a line starting with an asterisk:

```
* 1 4 80 11
```

This defines the rectangle which will surround the menu on the screen. The first number defines the column position for the left side of the rectangle, the second number is the row number of the top of the rectangle, the third is the column number of the right, and the last number is the row of the bottom. If no rectangle is specified, then the full screen (1, 1, 80, 24) will be assumed. In your program, when the menu is being displayed, it is possible to specify that no rectangle be drawn.

After the menu template, there is one resource for each of the fields. These resources describe the type and contents of each of the fields. For each of the fields in the menu template, there is an associated field resource. The first term in each field resource is the field identifier. This corresponds to the field number in the menu template.

The next term in the field resource tells the Menu Manager what type of field this is and what its default value and screen attributes are. In this example, field #0 is a PROMPT field. This is a field that the user cannot change. When the user is moving through the menu with the arrow keys, the cursor will not land on any PROMPT fields. The string in the quotes is the string that will be displayed in the menu for field #0. After that, there is an optional list of screen attributes that may be applied to the field. In this case, the text "Main Menu" will be displayed in BOLD on the terminal. You may list one or more of the following screen attributes separated by commas:

```
PLAIN
BOLD
UNDERLINE  or  UNDERSCORE
BLINK
INVERT  or  NEGATE
```

In a PROMPT field, PLAIN is the default.

Note: All keywords in menu resources must be in ALL CAPITALS. Default values in quotes may be mixed upper and lower case depending on how you want it to appear on the screen.

In field #2 we see an example of a STRING field. When the menu is first displayed, the default string "Default Name" will appear in the field. The user can then advance the cursor to the field and type a different string. Notice that we also have a list of optional screen attributes for this field: UNDERLINE. The default for a STRING field is BOLD and INVERT (reverse highlighted).

The next term in this string field is LEN=40. This sets the field length to 40 characters. The Menu Manager will prevent the user from typing beyond this limit. If no LEN term is specified, the length of the field will be whatever the length of the default string is.

Field #4 is an example of an INT (integer) field. Like the PROMPT and STRING terms, an INT field is followed by a default value and an optional list of screen attributes separated by commas. If no screen attributes are specified, the default for an INT field is BOLD and INVERT.

INT fields may also have an optional range checking term, RANGE. In this case, we are limiting the user's response to a range of 1 to 99. If the user types a number outside this range, the error message in the ERROR term will be displayed. The user must supply a number in the correct range in order to continue. If no ERROR term is supplied, the Menu Manager will display a generic error message showing the range. If no RANGE term is supplied, then no range checking will be performed.

Note: A field resource may be split across several lines as shown in fields #4 and #6.

Field #6 is an example of a FLOAT field. This is essentially the same as an INT field in format except that the user input is taken as a floating point (real) number. In field #6 there is also an optional NOTE term. When the user advances to this field with the arrow keys, this note will be displayed at the bottom of the menu (right above the bottom line in the menu rectangle.) You can use this to display any additional prompting information you want to give the user. Notice that both ERROR messages and NOTE messages appear at the bottom of the menu by default, but there is a way around that (more on that later).

At run time, when this menu is eventually displayed, it will look something like this:

```
┌──────────────────────────────────────────┐
│                Main Menu                 │
│                                          │
│ Your name:          Default Name         │
│ Your age:           21                   │
│ Your height:        68.0                 │
│                                          │
└──────────────────────────────────────────┘
```

The rectangle surrounding the menu will have its left edge in column 1, the top on row 4, the left in column 80 and the bottom in row 11. This leaves a blank line (line 10) at the bottom of the menu for the NOTE and ERROR messages.

Note: The spacing within a field resource is not critical. For example, you can put spaces around the equals sign or after the commas in a list of screen attributes. But there must be at least one space separating the different terms in the resource. In other words, after the #0 there must be at least one space, then after the <PROMPT="Main Menu",BOLD> there must also be at least one space. Note also that the spacing within the rectangle descriptor in the menu template is not critical as long as there is at least one space separating each number. For example, the following two resources will be parsed the same:

```
{ #3 STRING="Hello" LEN=10 }
{ #3  STRING = "Hello"   LEN = 10 }
```

However, these two lines are not quite the same:

```
{ #3 STRING="Hello" LEN=10 }
{ #3 STRING="HELLO" LEN=10 }
```

The difference is that the literal string in the second resource ("HELLO") is in all capitals. Whatever is within the quotes will appear on the screen exactly as typed. Apart from that, the two resources are the same.

3.5.3 Selection Menus:

In the previous section, 3.5.2, we saw how you might construct a menu that contains user changeable fields. This is also called a "screen form". But you may also want to display a list of items from which the user chooses one. The following is an example of the resources for such a menu:

Here is the menu template:

```
{ 5
            #0

                  #1
                  #2
                  #3
                  #4

      * 1 4 80 12
}
```

Here are the field resources:

```
{ #0  PROMPT="Choose one of the following:" }
{ #1  GROUP="First choice"  LEN=15 }
{ #2  GROUP="Second choice"  LEN=15 }
{ #3  GROUP="Third choice"  LEN=15 }
{ #4  GROUP="Special choice"  LEN=15
            SELECT=BOLD,BLINK,INVERT  DESELECT=BOLD }
```

In this example, the menu template is set up in the same way as in the previous section. Fields #1 through #4 are designated as GROUP fields. These fields can be selected or deselected, but not changed by the user. When the user moves the cursor with the arrow keys, the current GROUP field will become highlighted. The default highlighting is BOLD and INVERT. When a field is not selected, the default is PLAIN. Only one GROUP field may be selected at a time.

Notice that field #4 uses special highlighting. The SELECT term specifies the screen attributes for the field when it is selected (i.e. the user has moved to it with the arrow keys). The SPECIAL CHOICE will then be bolded, blinking, and the color will be inverted. The DESELECT term specifies the screen attributes for the field when it is not selected.

If you want the highlighting of all GROUP fields to be the same length, then the LEN term must be used. Otherwise, the length of each field will default to the actual length of the string in the quotes. Of course, you could also include trailing blanks inside the quotes to make all of the strings the same length.

At run time, when this menu is displayed, it will look something like this:

```
        Choose one of the following:

                First Choice
                Second Choice
                Special Choice
```

3.5.4 Resource File Details

The previous two sections gave an overall view of how resource files are set up. This section will discuss in more detail all of the possible terms that can go into field resources and the general format for both the menu template and the field resources. In the examples, angle brackets < > will be used to show that something is optional, and lower case italic letters or words such as 'n' or 'left' will be used to represent programmer supplied numbers.

The menu template resource is the first part of the resource file. It shows the locations of the fields in the menu. The general format for the menu template is as follows:

```
{ < s >
             #0            #1
     #2
     #3            #n ...

     < * left  top  right  bottom >
}
```

After the open brace, there is an optional parameter 's'. This number tells the Menu Manager that the next line in the template resource represents line 's' on the terminal screen. If this value is not supplied, the default is line 1. The column positions are exactly as they appear. For example, if the starting line were set to 5, and the '#' sign of the #0 field were in column 10, then the Menu Manager would set the position of field #0 to line 5, column 10.

After the open brace and the optional starting line number, there can be as many field position indicators as needed. The location of the field is determined by the position of the '#' sign relative to the starting line number. In the VAX C and Force C versions, the first field is #0. In the VAX Fortran version, the starting field number is #1. There can be as many fields as needed, but there should be no gaps in the numbering. The fields do not necessarily have to be in any order on the screen. When the user advances from field to field with the down arrow, the cursor will jump from field to field in the numbered order, and then wrap around back to #0. If a field is defined as a PROMPT, the cursor will skip over that field preventing the user from changing it.

At the end of the template, before the close brace, there is an optional rectangle definition. This shows the dimensions of the rectangle which will surround the menu when it is drawn on the terminal screen. The asterisk * indicates to the Menu Manager that the next four numbers are defining a rectangle. The four numbers represent the column of the left side of the rectangle, the row of the top of the rectangle, the column of the left, and the row of the bottom. If no rectangle definition is supplied, the default is the full screen (1, 1, 80, 24). Notice that the column of the asterisk is not critical and the spacing of the four numbers is not critical, as long as there is at least one space between each number.

Following the menu template resource, there are field resources. There should be one field resource for each field specified in the menu template. The general format for a field resource is as follows:

```
{ #n  FIELDTYPE  <term 1>  <term 2>  <term n>  ... }
```

After the open brace, the first term in the *field resource is the field number.* This corresponds to the number in the menu template. The field resources do not necessarily have to be in numeric order, but it makes the resource file easier to read.

The next term is the FIELDTYPE. This indicates what type of field we are defining. There are currently 5 field types:

PROMPT	-	Fixed field, not changeable by user.
INT	-	Integer number field, changeable by user.
FLOAT	-	Float (real) number field, user changeable.
STRING	-	String field, user changeable.
GROUP	-	User selectable, but not changeable field.

Any menu must have at least one field that is **not** a PROMPT. Otherwise, the user can never respond to the menu, and the program hangs. There can be any mixture of field types in a menu, although it is usually better to keep GROUP fields out of menus that have INT's, FLOAT's, or STRING's. It is perfectly legal to mix them, but it can appear confusing to the user.

The general format for one of the five FIELDTYPE terms is as follows:

```
FIELDTYPE = default <,attr1> <,attr2> <,attrn>
```

In the case of a PROMPT, STRING, or GROUP, the "default" is a string surrounded by double quotes. In the case of an INT or a FLOAT, the "default" is a number. There must be some default, even if it is a blank string or a zero. Notice that the spacing around the equals sign (and the commas) is not critical.

After the default value, there can be an optional list of screen attributes except in the case of a GROUP (more on that later). If screen attributes are used, there must be a comma after the default, and a comma between each attribute. One or more of the following screen attributes may be used:

```
PLAIN
BOLD
UNDERLINE  or  UNDERSCORE
BLINK
INVERT  or  NEGATE
```

If no screen attributes are used, a PROMPT will default to PLAIN. INT's, FLOAT's and STRING's will default to BOLD, INVERT. GROUP's are a special case. The screen attributes of a GROUP field depend on whether it is selected or not. The default for a selected GROUP field is BOLD, INVERT. The default for a deselected GROUP field is PLAIN. If these defaults are not what you want, then the SELECT and DESELECT terms can be included in the field resource in the following format:

```
SELECT = attr1 <,attr2> <,attr3> <,attrn>
DESELECT = attr1 <,attr2> <,attr3> <,attrn>
```

For example, if you want the GROUP field to be bold and blinking when is it selected, you would use the following:

```
SELECT = BOLD, BLINK
```

The SELECT and DESELECT terms may come anywhere after the GROUP term in the field resource.

There are also other special terms that apply only to certain field types. The RANGE term may be included in an INT or FLOAT field resource as follows:

```
RANGE = low, high
```

This will force the user to type a number between 'low' and 'high'. If the user inputs a number outside that range, a generic error message will be displayed at the bottom of the menu. If you wish to supply your own error message, the ERROR term may be included after the RANGE term as follows:

```
ERROR = "Error message" <,attr1> <,attr2> <,attrn>
```

The "Error message" can be any string whose length is less than the width of the menu rectangle. After the message, an optional list of screen attributes may be supplied. If no attributes are included, the default is BOLD, BLINK. Normally, the error message will appear on the last line of the menu, just above the bottom line of the rectangle. If this is not what you want, you can include the ERRLINE term after the ERROR term as follows:

```
ERRLINE = n
```

This specifies that the error message will appear on screen line 'n'.

On any field except a PROMPT, there may also be a NOTE. When the user advances to the field, any associated NOTE will be displayed at the bottom of the menu. The format of a NOTE term is similar to an ERROR term as follows:

```
NOTE = "Note message" <,attr1> <,attr2> <,attrn>
```

Like an ERROR term, a NOTE may have optional screen attributes. If none are supplied, the default is BOLD. Also like an ERROR, a NOTE will normally appear at the bottom of the menu, but that can be changed with the NOTELINE term as follows:

```
NOTELINE = n
```

This specifies that the note message will appear on screen line 'n'.

Any field may have a LEN term. This sets the length of the field to something other than the default. For example, if you have a STRING field whose default value is "Hello", that would normally default to length 5 since "Hello" has 5 letters. But, if you want to pad it with blanks out to 10 characters, you could include the LEN term as follows:

```
LEN = 10
```

This tells the Menu Manager to extend the field out to 10 characters and pad with blanks. If the field is already longer than 10, this will truncate the field to 10 characters. If you use the LEN term with a numeric field such as an INT or FLOAT, the field will be limited to the specified number of digits (including the decimal point, if applicable). For example, if you have an INT field whose default value is 42, the length would default to 2 since 42 has 2 digits. By using the LEN term, you could extend it to the maximum number of digits you would ever expect for that field.

Note: The Menu Manager will not allow the user to type beyond the current LEN limit of the field. Typing beyond that limit will cause the cursor to go to the next field.

There are four more terms that can be used with any field except a PROMPT. These are the LEFT, RIGHT, UP and DOWN terms. These can be used to control the movement of the cursor when the user presses the arrow keys. The format of these commands is as follows:

```
LEFT = n
RIGHT = n
UP = n
DOWN = n
```

The 'n' number specifies the field number to which the associated arrow key will advance. For example, if the resource for field #3 includes a RIGHT=5 term, then when the cursor is on field #3, pressing the right arrow will advance to field #5. This can be especially useful if your template has several columns of GROUP choices. You can set up the field links such that the left and right arrows will move from column to column, and the up and down arrows will move up and down in a column. Normally, when none of these terms are included, the down and right arrows will advance to the next non-PROMPT field numerically, and the up and left arrows will go back to the previous non-PROMPT field.

There is one other term that can be included in an INT, FLOAT or STRING field. That is the DEF term. (This is currently not implemented in the VAX Fortran version, but stay tuned!) The DEF term allows you to have a file containing default values for all of the menus. These defaults will override any default you have put directly into the resource. The values in this default file can also be updated at run time so that the next time the program is run, different defaults will be displayed in the menu. The format of a DEF term is as follows:

```
DEF = n  < ,UPDATE | NOUPDATE >
```

This says that line 'n' of the default file will contain the default value for this field. The next part of the term is an optional keyword, either UPDATE or NOUPDATE. If UPDATE is specified, then the default file will be updated with the new value the user types. The next time the menu is displayed, that new value will be the default. If

NOUPDATE is specified, then the default file will be used to get the default value, but it will not be updated after the user changes the field. If nothing is specified, UPDATE is the default.

The default file itself must be a stream oriented file **(not record oriented)** with all lines having the same number of characters. On the Force computer, such a file can be created very simply in the editor. The file would look like this:

```
Column 1
  |
  |
  42          Line 0.  Pad up to 'n' columns      ------> 000
  "Hello"     Line 1.  Comments go here.                001
  68.4        Line 2.  This is a parameter.             002

        ...  etc.  ...
```

Every line must have the same number of characters in it. The default value is the first thing on the line. Anything after it is considered a comment. The line must be padded with blanks (or anything) up to the desired fixed length. It is convenient, though not required, to end each line with a line number. In this example, line 0 has the parameter value 42. If a field in a menu had a DEF=0 term, the Menu Manager would put 42 into the field before displaying the menu. Then, if it was updatable, and the user typed 55, that number would be written into the file in place of the 42. The next time the menu is displayed, 55 would be used as the default value.

On the VAX, there is a slight complication. When you create a file by editing it, it automatically becomes record oriented. In order to convert it to a stream oriented file, you would have to write a small C program that simply opens the record oriented file, reads it byte by byte and writes it out to another file byte by byte. The resulting output file will be stream oriented.

3.5.5 Using Menus in a Program

Now you have created your resource file for the menu. How do you actually display and handle the menu in your program? The first thing your program must do is get the resources from the file. Then, it must display the menu. And finally, it must look at the user responses to determine what to do next.

On the following pages, two examples are shown. The first shows the 'C' interface, and the second shows the Fortran interface. Both programs represent essentially the same code. The example uses the resource file we created in the "Selection Menus" section of this document, so it shows how to handle a menu with selections. (The next section will show how to handle "screen form" type menus.) Following the two programs is a detailed discussion:

```
/*** 'C' version: ***/

#include <stdio.h>
#include <menus.h>
#include <functions.h>

#define  MAX  5     /* Max number of fields in menu.   */


main ()
{
   int          status;          /* Function return status.    */
   MenuRec TheMenu[MAX];         /* The menu data structure.   */
   Rectangle    TheBox;          /* Rectangle around menu.     */
   int          NumFields;       /* Number of fields in menu.  */
   int          MaxField;        /* Highest field number.      */
   int          field;           /* Current menu field number. */


   /* Initialize the "virtual keyboard".  Do this once    */
   /* in the beginning of all programs that use menus:    */
   INITVKEYS (&status);
   if ( ! status )
   {
      printf ("Error setting up virtual keyboard!\n");
      return;
   }

   /* Read in the resource file and parse it: */
   status = GetMenuF ("selectmenu.rsc", TheMenu,
            &TheBox, &NumFields, &MaxField, NULL, 0);
   if ( ! status )
   {
      printf ("Error getting resources!\n");
      return;
   }

   /* Display the menu and let the user make a selection: */
   field = 0;  /* Start with cursor on field #0. */
   status = DoMenu (TheMenu, TheBox, MaxField, &field, BOLD,
            DEFAULTMENU, "helpfile.hlp", NULL, 0);

   /* Now show the user what was chosen from the menu: */
   CLEARSCREEN;  HOMECURSOR;
   printf ("You chose field number %d.\n", field);

   /* Clean up screen and exit: */
   Quitting ();
}


C  *** Fortran version:  ***

   PROGRAM  TESTMENU

   INCLUDE      'MENUS.FH'
```

```
      INCLUDE     'FUNCTIONS.FH'
      PARAMETER   MAX = 5

      INTEGER     STATUS                    ! Subroutine return status.
      RECORD /MENUREC/  THEMENU(MAX)              ! The menu data structure.
      RECORD /RECT/ THEBOX                  ! Rectangle around menu.
      INTEGER     NUMFIELDS                 ! Number of fields in menu.
      INTEGER     MAXFIELD                  ! Highest field number.
      INTEGER     FIELD                     ! Current menu field number.


C     Initialize the "virtual keyboard".  Do this once
C     in the beginning of all programs that use menus:
      CALL INITVKEYS (STATUS)
      IF (STATUS .EQ. 0) THEN
         WRITE (5, *) 'Error setting up virtual keyboard!'
         STOP
      ENDIF

C     Read in the resource file and parse it:
      STATUS = GETMENUF ('selectmenu.rsc', 10, THEMENU, THEBOX,
     +                       NUMFIELDS, MAXFIELD, 0)
      IF (STATUS .EQ. 0) THEN
         WRITE (5, *) 'Error getting resources!'
         STOP
      ENDIF

C     Display the menu and let the user make a selection:
      FIELD = 1          ! Start with cursor on field #1.
      CALL DOMENU (THEMENU, THEBOX, MAXFIELD, FIELD, BOLD,
     +                    DEFAULTMENU, 'helpfile.hlp', STATUS)

C     Now show the user what was chosen from the menu:
      CALL SCREENCONTROL (CLEARSCREEN // HOMECURSOR)
      WRITE (5, 20) FIELD
   20 FORMAT (1X, 'You chose field number ', I3)

C     Clean up screen and exit:
      CALL QUITTING

      END ! TESTMENU
```

How does this all work? In the beginning of both versions of the program are some declarations. Your program must include "menus.h" and "functions.h" (or "menus.fh" and "functions.fh" for Fortran). These files contains data definitions, constants, and function declarations. The constant MAX is defined as the number of fields in the menu, in this case 5.

Next are the data declarations. In both languages, some special data types are used. The menus are declared as an array of data structures. The structures "MenuRec" and "MENUREC" are defined in "menus.h" and "menus.fh" respectively. The size of the array is the number of fields in the menu. The number can be bigger to allow for expansion, but not smaller. Notice there is also a special data type for the menu rectangle. This structure consists of four integer fields representing the four sides of the rectangle.

The first line of executable code in both versions is a call to the routine "INITVKEYS". This routine initializes the "virtual keyboard" on the VAX, and some other initialization on the Force. (See the VAX documentation on "virtual keyboards.") This routine must be called once before any other menu calls. It is usually convenient to call it at the very beginning of the program. If it returns a status of zero, something went wrong and your program cannot continue. However, if this should happen, contact the software manager.

The next section of code reads in the resource file, parses it, and initializes the fields in the menu data structure plus the associated variables. Notice in the 'C' version, the last two parameters of GetMenuF are set to NULL and 0. If you are using a default parameter file, these would be set to the file pointer and the number of characters per line in the default parameter file. Notice also that in the Fortran version, the second parameter is set to 10. This is the logical unit number we want the Menu Manager to use for reading in the resource file. Also, the last parameter is set to 0. This is the logical unit number for the default parameter file. At present, this is not implemented in the Fortran version. Like INITVKEYS, GetMenuF will return a zero status if something went wrong such as file not found.

In the next section, we are actually displaying the menu. Notice that we are setting "field" to 0 (or 1 in the Fortran version) before we call DoMenu. This value gets passed to DoMenu and tells the Menu Manager where to put the cursor initially. When the menu is displayed with DoMenu, the cursor will start on that field unless it is a PROMPT field. In that case, the cursor will advance to the next non-PROMPT field.

DoMenu handles all of the user input. It moves the cursor from field to field and handles any user changes to INT, FLOAT and STRING fields. (In this example, the fields are all GROUP's and PROMPT's.) Notice that in both languages, the fifth parameter is BOLD. This is the screen attribute of the menu rectangle. This parameter may be one or the sum of any of the following:

```
PLAIN
BOLD
UNDERLINE  or  UNDERSCORE
BLINK
INVERT  or  NEGATE
NODRAWBOX
NODRAWTOP
NODRAWBOT
```

If NODRAWBOX is used, the box is not drawn at all. If NODRAWTOP or NODRAWBOT are added to the parameter, then the top and/or bottom of the rectangle are not drawn. You may want to have a section of the screen that never changes, for example.

In both languages, the next parameter in DoMenu is the menu mode. This can be one or the sum of any of the following:

```
DEFAULTMENU
NOHANDLEMENU
NOPRECLEAR
PRECLEARTOP
NODRAWMENU
```

If DEFAULTMENU is used, the screen will be cleared from the first line of the menu to the end of the screen before drawing the menu, then the menu will be drawn and user input will be handled. If NOHANDLEMENU is used, the menu will be drawn, but then control will return directly to the calling routine before any user input is handled. NOPRECLEAR will draw the menu without doing any screen clearing first. PRECLEARTOP will clear from the last line of the menu up to the top of the screen (as opposed to the first line of the menu down to the bottom of the screen) before drawing the menu. And NODRAWMENU will just go directly to handling the user responses without drawing the menu. You might use this if the menu has already been drawn with NOHANDLEMENU, for example.

The next parameter in both languages is the name of the optional help file. Any menu can have a help file associated with it. The format of the file is just a text file with no more than 'n' characters per line, where 'n' is the width of the menu rectangle minus 4. When the user presses the HELP key or PF2, this help file will be displayed within the menu rectangle, one section at a time. If no help file is to be used, pass a NULL string to DoMenu.

The next two parameters in the 'C' version are for default parameter files. The first parameter is the actual file pointer. The default parameter file should be opened before calling DoMenu. The next parameter is the number of characters per line in the default parameter file. If no parameter file is to be used, pass NULL for the file pointer and zero for the number of characters per line. This is not yet implemented in the Fortran version, but it may be implemented in a future version.

In both languages, the returned "status" indicates how the menu was dismissed by the user. This status can be compared against the following predefined constants:

```
MENUQUIT
MENUCANCEL
MENUOKAY
```

If the return value is MENUQUIT, it means the user selected the special program quitting option. This is usually a "hidden" option. When the user types

"/Q<RETURN>" (slash-Q-return) in any field, this will signal the calling routine that the user wants to get completely out of the program. Your calling routine would then have to check for MENUQUIT in order to respond properly. But since this is normally not known to the user, you may choose simply to ignore MENUQUIT.

If the return status value is MENUCANCEL, it means that the user pressed PF4 (or the MENU key on the Force) and wants to cancel out of this menu and go back up to the previous level. Again, your program would have to check for this in order to respond properly.

If MENUOKAY is returned, the user completed the changes or choices by pressing ENTER (also DO on the VAX). Your program can then proceed as necessary.

When we return from DoMenu, the "field" will be set to the last field that the user selected before pressing ENTER. In our example, we are displaying that number for verification. In both versions we are clearing the display and moving the cursor to the home position (upper left) before showing the results. In 'C', the CLEARSCREEN and HOMECURSOR commands are predefined in "menus.h" as macros which send out the appropriate escape sequences to the terminal. In Fortran, the subroutine SCREENCONTROL is called to do this sort of screen handling. The parameter to SCREENCONTROL is a control string of escape sequences. These strings are predefined in "menus.fh", and any number of them may be concatenated together to be passed to SCREENCONTROL.

The last thing in both programs is a call to QUITTING. This just resets the screen parameters to make sure your program doesn't exit leaving the screen in some strange mode.

3.5.6 More about Using Menus in a Program

In the previous section, we saw how you would handle a menu that has several choices from which the user chooses one. The returned "field" indicates which field was selected by the user.

It is also possible to handle menus where the user types responses into the fields. The basic program structure would be the same as in the previous section. The difference is in how the returned values are handled. When DoMenu returns to the calling routine, you can examine the values in TheMenu data structure directly. For example, if field #2 were a STRING field, the string value would be accessed like this:

```
The Menu[2].value        /* 'C' version.  */
THE MENU(2).VALUE        !  Fortran version.
```

For INT and FLOAT fields, the string value can be accessed as above, but the numeric value can also be accessed like this:

```
TheMenu[2].intval         /* 'C' version.  */
TheMenu[2].floatval
THEMENU(2).INTVAL         !  Fortran version.
THEMENU(2).FLOATVAL
```

Individual characters in the string value can be accessed like this:

```
TheMenu[2].value(n)       /* 'C' version.  */
THEMENU(2).VALUE(N:N)     !  Fortran version.
```

Upon returning from DoMenu, the calling routine may examine these values and take whatever action is appropriate. It may also be necessary to redisplay the menu if the user was in error. It is therefore convenient to structure your menu handling in a loop like this:

```
/*** C version:  ***/

    ...

    /* Set up the menu parameters outside of the loop:  */
    moremenu = 1;         /* 0 = done with menu.  */
    field = 0;            /* Start with cursor on field #0. */
    mode = DEFAULTMENU;   /* First time, handle normally.   */

    while (moremenu)
    {
        /* Display the menu and let the user make a selection: */
        status = DoMenu (TheMenu, TheBox, MaxField, &field, BOLD,
                 mode, "helpfile.hlp", NULL, 0);

        /* If the user typed "X" in field #2, force retry: */
        if (TheMenu[2].value(0) == 'X')
        {
             row = 23;
             col = 1;
             MOVECURSOR (row, col);
             printf ("Please try again ...");
             IDLE (2);
             mode = NODRAWMENU;
        }

        else      /* Response was okay:  */
        {
             moremenu = 0;   /* Force loop exit. */
             /* ... Handle good user responses here ... */
```

```
         }
     }

     ...


C    *** Fortran version ***

     ...

C    Set up the menu parameters outside of the loop:
     MOREMENU = .TRUE.     ! .FALSE. = done with menu.
     FIELD = 1             ! Start with cursor on field #1.
     MODE = DEFAULTMENU    ! First time, handle normally.

     DO WHILE (MOREMENU)

C        Display the menu and let the user make a selection:
         CALL DOMENU (THEMENU, THEBOX, MAXFIELD, FIELD, BOLD,
    +                 MODE, 'helpfile.hlp', STATUS)

C        If the user typed 'X' in field #2, force retry:
         IF (THEMENU(2).VALUE(1:1) .EQ. 'X') THEN
              ROW = 23
              COL = 1
              CALL MOVECURSOR (ROW, COL)
              WRITE (5, *) 'Please try again ...'
              CALL IDLE (2)
              MODE = NODRAWMENU;

         ELSE     ! Response was okay:
              MOREMENU = .FALSE.
              ! ... Handle good responses here ...

         ENDIF

     ENDDO
     ...
```

First of all, we are setting up certain loop parameters before we actually get into the loop. Instead of putting DEFAULTMENU directly into the DoMenu call, we are using a variable called 'mode'. This will allow us to change the menu drawing mode the next time through the menu.

Inside the loop, after DoMenu is called, we are checking to see if the user typed 'X' in field #2. If so, we want to display an error message, and go back into DoMenu to force the user to try again. Notice we are moving the cursor out of the menu by using the macro MOVECURSOR in 'C' or the subroutine MOVECURSOR in Fortran. Then, we can display the error message without laying it on top of the menu. The call to IDLE just forces the program to wait for a certain number of seconds to pass before continuing. Then, we set the mode to NODRAWMENU. Now, when we go back to the top of the loop, DoMenu will start handling the user responses without redrawing the menu. Redrawing is not necessary at this point since we have not yet erased it.

If the user did not type 'X' in field #2, we are assuming that it is okay. We can therefore do whatever processing we want. Notice that we set the 'moremenu' flag to 0 (or in the case of Fortran, MOREMENU is set to .FALSE.) so that we will exit from the loop.

Selection menus can be handled similarly. Suppose we have the case where there are 4 GROUP fields in the menu resource. To handle the user response we could structure the loop like this:

```
/*** C version:  ***/

    ...

    /* Set up the menu parameters outside of the loop:  */
    moremenu = 1;        /* 0 = done with menu.  */
    field = 0;           /* Start with cursor on field #0. */
    mode = DEFAULTMENU;  /* First time, handle normally.   */

    while (moremenu)
    {
        /* Display the menu and let the user make a selection: */
        status = DoMenu (TheMenu, TheBox, MaxField, &field, BOLD,
                 mode, "helpfile.hlp", NULL, 0);

        /* Check if the user canceled this menu: */
        if (status == MENUCANCEL)
        {
            moremenu = 0;        /* Force loop exit. */
        }

        /* User response was okay.  Check what was selected:  */
        else if (field == 1)
        {
            /* ... Handle response to field #1 selection ... */
        }
        else if (field == 2)
        {
            /* ... Handle response to field #2 selection ... */
        }
        else
        {
            /* ... etc. ...  /
        }
    }

    ...

/***  (A switch/case statement could also be used above.)  ***/
```

```
C   *** Fortran version ***

    ...

C   Set up the menu parameters outside of the loop:
    MOREMENU = .TRUE.     ! .FALSE. = done with menu.
    FIELD = 1             ! Start with cursor on field #1.
    MODE = DEFAULTMENU    ! First time, handle normally.

    DO WHILE (MOREMENU)

C       Display the menu and let the user make a selection:
        CALL DOMENU (THEMENU, THEBOX, MAXFIELD, FIELD, BOLD,
   +                 MODE, 'helpfile.hlp', STATUS)

C       Check if the user canceled this menu:
        IF (STATUS .EQ. MENUCANCEL) THEN
             MOREMENU = .FALSE.

C       User response was okay.  Check what was selected:
        ELSE IF (FIELD .EQ. 1) THEN
C            ... Handle response to field #1 selection ...
        ELSE IF (FIELD .EQ. 2) THEN
C            ... Handle response to field #2 selection ...
        ELSE
C            ... etc. ...
        ENDIF

    ENDDO

    ...
```

This method of structuring the loops makes it very easy to nest menus. For example, suppose when the user selects field #2 we want to put up another menu. We would just check for field #2 in the loop and then call another subroutine which puts up a menu in the same manner. When the user exits from that menu, the subroutine will return and loop back to display the first menu again. The only difference is that we would have to make sure that the 'mode' variable is set to DEFAULTMENU to force the menu to be redrawn.

3.5.7 Related Topics

In addition to the subroutines, functions and methods shown in the previous sections, there are other things that can be done with the Menu Manager that are not necessarily directly related to menus. For example, there are many predefined screen controlling macros to manipulate the cursor, screen mode, and other things. In 'C', these macros are used by themselves followed by a semicolon. In Fortran, they are passed to SCREENCONTROL individually, or in a concatenated string. Here is a partial list of macros available:

GRAPHCHARS	- Puts screen into special characters mode.
ASCIICHARS	- Puts screen back into normal characters.
CLEARSCREEN	- Clear whole screen.
CLEARTOEND	- Clear from cursor to end of screen.
CLEARTOSTART	- Clear from cursor to top of screen.
CLEARLINE	- Clear line cursor is on.
HOMECURSOR	- Move cursor to home position (upper left).
BOTTOMCURSOR	- Move cursor to row 24, column 1.
LEVEL1	- Put terminal into VT-100 mode.
LEVEL2	- Put terminal into VT-220, 7-bit mode.
CURSOROFF	- Turn cursor off.
CURSORON	- Turn cursor on.
SMOOTHSCROLL	- Put terminal into smooth scroll mode.
JUMPSCROLL	- Put terminal into jump scroll mode.

There are also screen drawing subroutines available. These routines can be used to draw lines and boxes, draw text inside a box, prompt the user for input in a box, scroll information inside a box, and several other things. And there are related routines which do things like parse tokens from strings, and handle "virtual keyboard" input from the user. The following section describes some of the more useful of these routines and shows the interface for both languages:

MOVECURSOR: Move the cursor to a screen position specified by the row and column. The 'C' version is a macro.

C:

```
int      row, column;
MOVECURSOR (row, column);
```

Fortran:

```
INTEGER      ROW, COLUMN
CALL MOVECURSOR (ROW, COLUMN)
```

SETSCROLL: Set the scrolling region on the screen. If a line is printed at the bottom of the region, the lines will scroll up. The 'C' version is a macro.

C:

```
int      top, bottom;
SETSCROLL (top, bottom);
```

Fortran:

```
INTEGER      TOP, BOTTOM
CALL SETSCROLL (TOP, BOTTOM)
```

SetTextAttr: Set the current text attributes for subsequent printing. The attributes may be any one or the sum of any of the following predefined constants:

```
PLAIN
BOLD
UNDERLINE  or  UNDERSCORE
```

```
BLINK
INVERT  or  NEGATE
```

C:

```
int      attributes;
SetTextAttr (attributes);
```

Fortran:

```
INTEGER      ATTRIBUTES
CALL SETTEXTATTR (ATTRIBUTES)
```

DrawRect: Draw a rectangle in graphic line segment characters. The rectangle data structure is predefined in "menus.h" or "menus.fh" as follows:

C:

```
typedef struct
{
     int      left, top, right, bottom;
} Rectangle;
```

Fortran:

```
STRUCTURE /RECTANGLE/
     INTEGER     LEFT, TOP, RIGHT, BOTTOM
END STRUCTURE
```

The following example shows how to draw a rectangle with the left edge in column 1, the top in row 3, the right in column 80 and the bottom in row 15. (See also SetRect.):

C:

```
Rectangle    TheRect;
SetRect (&TheRect, 1, 3, 80, 15);
DrawRect (TheRect);
```

Fortran:

```
RECORD /RECTANGLE/   THERECT
CALL SETRECT (THERECT, 1, 3, 80, 15)
CALL DRAWRECT (THERECT)
```

SetRect: Set the four coordinates of a rectangle data structure. This merely saves the step of having to assign each one in a separate statement. See the above example for DrawRect.

DrawVLine: Draw a vertical line with graphic line segment characters. The data structures for a vertical line are predefined in "menus.h" or "menus.fh" as follows:

C:

```
typedef struct
{
     int      top, bottom, column;
} VLine;
```

Fortran:

```
STRUCTURE /VLINE/
     INTEGER     TOP, BOTTOM, COLUMN;
END STRUCTURE
```

The following example draws a vertical line in column 12 from row 5 to row 20:

C:

```
VLine   TheLine;
TheLine.top = 5;
TheLine.bottom = 20;
TheLine.column = 12;
DrawVLine (TheLine);
```

Fortran:

```
RECORD /VLINE/   THELINE
THELINE.TOP = 5
THELINE.BOTTOM = 20
THELINE.COLUMN = 12
CALL DRAWVLINE (THELINE)
```

DrawHLine: This is identical to the above DrawHLine except that it draws a Horizontal line. The data structures are as follows:

C:

```
typedef struct
{
     int      left, right, row;
} HLine;
```

Fortran:

```
STRUCTURE /HLINE/
     INTEGER     LEFT, RIGHT, ROW;
END STRUCTURE
```

The routine would be used as above.

DrawText: Draw text within a rectangular clipping area. The text is wrapped (but not word wrapped) within the rectangle, and clipped if necessary. The string can be declared to any length. The text attributes can be one or the sum of the attributes listed under SetTextAttr (above).

C:

```
Rectangle    TheRect;
char         TheString[80];
int          TextAttrs;
DrawText (TheRect, TheString, TextAttrs);
```

Fortran:

```
RECORD /RECTANGLE/   THERECT
CHARACTER*80         THESTRING
INTEGER              TEXTATTRS
CALL DRAWTEXT (THERECT, THESTRING, TEXTATTRS)
```

TextBox[C]: Draw text within a visible rectangle. The text and the box may have different screen attributes. TextBoxC is identical except the rectangle is cleared out first.

C:

```
Rectangle    TheRect;
char         TheString[80];
int          TextAttrs, BoxAttrs;
TextBox (TheRect, TheString, TextAttrs, BoxAttrs)
```

Fortran:

```
RECORD /RECTANGLE/   THERECT
CHARACTER*80 THESTRING
INTEGER      TEXTATTRS, BOXATTRS
```

```
CALL TEXTBOX (THERECT, THESTRING, TEXTATTRS,
                             BOXATTRS)
```

FillRect: Fill a rectangular area with a repeated graphic character. Some of the predefined graphic characters are as follows:

BLANK	- A graphic blank.
GRAY	- A gray halftoned cursor-type block.
BRCORNER	- Bottom right rectangle corner.
TRCORNER	- Top right rectangle corner.
BLCORNER	- Bottom left rectangle corner.
TLCORNER	- Top left rectangle corner.
CROSS	- Line segment intersection.
LEFTT	- A left 'T' intersection.
RIGHTT	- A right 'T' intersection.
TOPT	- A top 'T' intersection.
BOTTOMT	- A bottom 'T' intersection.
VBAR	- Vertical line segment.
HBAR	- Horizontal line segment.
DOT	- Center dot.

C:

```
Rectangle    TheRect;
char         GraphChar;
GraphChar = GRAY;
FillRect (TheRect, GraphChar);  /* OR: */
FillRect (TheRect, GRAY);
```

Fortran:

```
RECORD /RECTANGLE/  THERECT
CHARACTER*1         GRAPHCHAR
GRAPHCHAR = GRAY
CALL FILLRECT (THERECT, GRAPHCHAR)  ! OR:
CALL FILLRECT (THERECT, GRAY)
```

ClearInside: Clear the inside of a rectangular area. This could also be accomplished by calling FillRect with BLANK.

C:

```
Rectangle    TheRect;
ClearInside (TheRect);
```

Fortran:

```
RECORD /RECTANGLE/  THERECT
CALL CLEARINSIDE (THERECT)
```

Dialog[C]: Put up a rectangle with a prompting message and wait for the user to type a response. The prompt, the user response, and the box can all have different screen attributes. Also, the prompt can be a string constant passed directly in the subroutine call. DialogC is identical except that the rectangle is cleared first.

C:

```
Rectangle    TheRect;
char         Prompt[80];
char         UserResponse[80];
```

```
                int          PromptAttrs, BoxAttrs, UserAttrs;
                Dialog (TheRect, Prompt, UserResponse,
                             PromptAttrs, BoxAttrs, UserAttrs);
```

Fortran:

```
                RECORD /RECTANGLE/  THERECT
                CHARACTER*80 PROMPT
                CHARACTER*80 USERRESPONSE
                INTEGER      PROMPTATTRS, BOXATTRS, USERATTRS
                DIALOG (THERECT, PROMPT, USERRESPONSE,
                             PROMPTATTRS, BOXATTRS, USERATTRS)
```

DialogT[C]: Similar to the above Dialog, except that the number of characters in the user response can be limited by the calling routine and the user may press non-alpha-numeric keys such as PF1-PF4, ENTER, arrow keys, etc. as string terminators. The termination character that is returned may be compared against the following predefined constants:

CR_KEY	- Carriage return.
EOS_KEY	- End Of String (character limit).
UP_KEY	- Up arrow.
DOWN_KEY	- Down arrow.
LEFT_KEY	- Left arrow.
RIGHT_KEY	- Right arrow.
HELP_KEY	- Help key on VT-220 or higher.
DO_KEY	- Do key (or MENU on Force).
PF1_KEY	- PF1 key on keypad.
PF2_KEY	- PF2 key on keypad.
PF3_KEY	- PF3 key on keypad.
PF4_KEY	- PF4 key on keypad.
ENTER_KEY	- Enter key on keypad.

Note: DialogTC is identical except that the rectangular area is cleared out first.

C:

```
                Rectangle    TheRect;
                char         Prompt[80];
                char         UserResponse[80];
                int          PromptAttrs, BoxAttrs, UserAttrs;
                int          NumChars;
                short int    Terminator;
                DialogT (TheRect, Prompt, UserResponse,
                             PromptAttrs, BoxAttrs, UserAttrs
                             NumChars, Terminator);
```

Fortran:

```
                RECORD /RECTANGLE/  THERECT
                CHARACTER*80 PROMPT
                CHARACTER*80 USERRESPONSE
                INTEGER      PROMPTATTRS, BOXATTRS, USERATTRS
                INTEGER      NUMCHARS
                INTEGER*2        TERMINATOR
                DIALOGT (THERECT, PROMPT, USERRESPONSE,
                             PROMPTATTRS, BOXATTRS, USERATTRS
```

```
                    NUMCHARS, TERMINATOR)
```

ScrollBox: Put up a rectangle in which messages can be scrolled independently of the rest of the screen. More than one scroll box can be on the screen at once, but they cannot be on the same lines. In other words, one can be above another, but not next to another. This routine is called first to initialize the box. Subsequent messages are scrolled with ScrollText and ScrollDialog.

C:

```
Rectangle    TheRect;
char         InitMessage[80];
int          TextAttrs, BoxAttrs;
ScrollBox (TheRect, InitMessage, TextAttrs,
                              BoxAttrs);
```

Fortran:

```
RECORD /RECTANGLE/  THERECT
CHARACTER*80 INITMESSAGE
INTEGER      TEXTATTRS, BOXATTRS
CALL SCROLLBOX (THERECT, INITMESSAGE, TEXTATTRS,
                              BOXATTRS)
```

ScrollText: Scroll a line of text within a scroll box. ScrollBox must have been called first to initialize the scroll box. The text will be drawn at the bottom of the scroll box and the rest of the text will be scrolled up.

C:

```
Rectangle    TheRect;
char         TheText[80];
int          TextAttrs, BoxAttrs;
ScrollText (TheRect, TheText, TextAttrs,
                              BoxAttrs);
```

Fortran:

```
RECORD /RECTANGLE/  THERECT
CHARACTER*80 THETEXT
INTEGER      TEXTATTRS, BOXATTRS
CALL SCROLLTEXT (THERECT, THETEXT, TEXTATTRS,
                              BOXATTRS)
```

ScrollDialog: Scroll a prompt message in a scroll box and wait for a user response. ScrollBox must have been called first to initialize the scroll box.

C:

```
Rectangle    TheRect;
char         Prompt[80];
char         Response[80];
int          PromptAttrs, BoxAttrs, UserAttrs;
ScrollDialog (TheRect, Prompt, Response,
                 PromptAttrs, BoxAttrs, UserAttrs);
```

Fortran:

```
RECORD /RECTANGLE/  THERECT
CHARACTER*80 PROMPT
CHARACTER*80 RESPONSE
INTEGER      PROMPTATTRS, BOXATTRS, USERATTRS
```

```
                    CALL SCROLLDIALOG (THERECT, PROMPT, RESPONSE,
                                  PROMPTATTRS, BOXATTRS, USERATTRS)
```

DrawWindowTitle: Draw a title on a scroll box in one of several styles. If StyleChar is 0, the title will be left justified on the top line of the scroll box rectangle. If StyleChar is 1, the title will be centered. If StyleChar is 2, the title will be right justified. Any other StyleChar is taken to be a fill pattern to use in a title bar. A title bar is drawn above the rectangle to look sort of like a MacIntosh window. DrawWindowTitle should be called right after ScrollBox. The fill pattern characters can be any of the graphic characters outlined in FillRect. The rectangle is the rectangle above which the title will be drawn (the scroll box rectangle.)

```
    C:              Rectangle    TheRect;
                    char         Title[80];
                    char         StyleChar;
                    int          TitleAttrs, BoxAttrs, BarAttrs;
                    DrawWindowTitle (TheRect, Title, StyleChar,
                                  TitleAttrs, BoxAttrs, BarAttrs);

    Fortran:        RECORD /RECTANGLE/   THERECT
                    CHARACTER*80         TITLE
                    CHARACTER*1          STYLECHAR
                    INTEGER      TITLEATTRS, BOXATTRS, BARATTRS
                    CALL DRAWWINDOWTITLE (THERECT, TITLE, STYLECHAR,
                                  TITLEATTRS, BOXATTRS, BARATTRS)
```

GetLine: Get a line of text from a regular ASCII text file. This may be useful for parsing text along with GetToken. In Fortran, LINELEN returns the number of characters in the string not including the trailing blanks. In 'C', this can be accomplished by using the strlen function on the returned string. (LEN in Fortran includes the trailing blanks.) If EOF is reached, status will be 0.

```
    C:              FILE *thefile;
                    char    TheString[80];
                    int     status;
                    status = GetLine (thefile, TheString);

    Fortran:        INTEGER          LOGICALUNIT
                    CHARACTER*80     THELINE
                    INTEGER          LINELEN
                    INTEGER          STATUS
                    STATUS = GETLINE (LOGICALUNIT, THELINE, LINELEN)
```

GetToken: Parse a token out of a string. A token is a string of visible characters delimited by a space, or a comma, or enclosed by double quotes. For example:

```
     Hello there.  TREE=MAPLE  "two words"  a,b
```

In the above line, the tokens are: "Hello", "there.", "TREE", "=", "MAPLE", "two words", "a", ",", "b". Notice that the quotes are not actually part of the token. Also notice that the visible delimiters (comma and quote) are tokens by themselves. Blanks are not tokens, but are token delimiters. The token "two words" is a single token because it is surrounded by double quotes in the original string. Without the quotes, it would be two tokens. GetToken looks at the input string starting at the character position NextChar and returns the next token. The function returns 0 if no tokens are found, or the length of the next token, if found. In Fortran, GetToken is declared as an INTEGER FUNCTION. Upon returning, NextChar will be incremented to the end of the token. By calling GetToken repeatedly in a loop, you can parse out all of the tokens in a string.

C:

```
char InputString[80];     /* Can be longer. */
int      NextChar;
char     TheToken[80];
int      TokenLength;
TokenLength = GetToken (InputString, &NextChar,
                        TheToken);
```

Fortran:

```
CHARACTER*80 INPUTSTRING  ! Or longer.
INTEGER      NEXTCHAR
CHARACTER*80 THETOKEN
INTEGER      TOKENLENGTH
TOKENLENGTH = GETTOKEN (INPUTSTRING, NEXTCHAR,
                        THETOKEN)
```

INITVKEYS: Initialize the "virtual keyboard". This must be done once in the beginning of any program that uses menus or the "GETVSTRING" and "GETVCHAR" functions. The virtual keyboard allows your program to respond to single keystrokes or other sequences not necessarily terminated with a RETURN. For example, you may want to allow arrow keys to by typed by the user. (The return status is not currently being used.)

C:

```
int     status;
INITVKEYS (&status);
```

Fortran:

```
INTEGER STATUS
CALL INITVKEYS (STATUS)
```

GETVSTRING: Get a string input from the virtual keyboard. GETVSTRING will not accept any keys beyond the maximum selected with the maxchars parameter. The terminator code indicates how the string was terminated by the user. For example, if the user typed the string and then pressed RETURN, the terminator code would be CR_KEY. If the user typed the maximum number of characters, the input would be automatically terminated with EOS_KEY. See DialogT for a list of termination codes.

C:

```
char     TheString[80];    /* Typed by user. */
int      NumChars;         /* Num chars in string. */
short int    Terminator;
int      MaxChars;
GETVSTRING (TheString, &NumChars, &Terminator,
                               &MaxChars);
```

Fortran:

```
CHARACTER*80 THESTRING
INTEGER      NUMCHARS
INTEGER*2        TERMINATOR
INTEGER      MAXCHARS
CALL GETVSTRING (THESTRING, NUMCHARS, TERMINATOR,MAXCHARS)
```

GETVCHAR: Get a single character input from the virtual keyboard. This is the same as calling GETVSTRING with maxchar set to 1. If mode is set to 1, the character the user types will be echoed to the screen. If it is 0, it will not be echoed.

C:

```
char TheChar; /* Typed by user. */
short int    Terminator;
int      mode;
GETVCHAR (TheString, &Terminator, &mode);
```

Fortran:

```
CHARACTER*1      THECHAR
INTEGER*2            TERMINATOR
INTEGER          MODE
CALL GETVCHAR (THESTRING, TERMINATOR, MODE)
```

3.6 Recipe Coder / Editor / Analyzer Program Details

3.6.1 Introduction

The Recipe CEA is an interactive program on the VAX for creating and editing coded recipe files and then analyzing them for nutritional composition. This document describes some of the finer details on the program. It is intended more for the programmer, not the user. For details on how to use the program, see the User's Guide. See also the Programmer's Guide.

3.6.2 Program Files

The Recipe CEA consists of several Fortran source code files, the Menu Manager source code files, the link file, the resource files (for menus), the help files, and a couple other data files. The following is a list of all files used with a brief description of each:

RECIPE SOURCE CODE

RECIPE.FOR	- Main menu, coding functions, etc.
INGRED.FOR	- Prompt for ingredients, units, etc.
RECEDIT.FOR	- Recipe editing functions.
RECIPE_ANA.FOR	- Main analysis routines.
SET_UP.FOR	- Analysis menu and initialization.
GET_YIELDS.FOR	- Parse fat/water/yield changes in analysis.
REPORT.FOR	- Output analysis report.
INGRED_ANA.FOR	- Ingredient analysis & retentions.
NEW_RECIPE.FOR	- Handle first line of a recipe.
PROCESS_DEL.FOR	- Sub-recipe closing, adjust yields, etc.

MENUS & GRAPHICS SOURCE CODE

MENUS.FOR	- Menu handling functions.
GRAPHICS.FOR	- Screen line graphics (boxes, etc.)
UTILS.FOR	- Utilities (get tokens, etc.)
DATEUTILS.FOR	- Date & time functions.

RESOURCE FILES

RECMENU1.RSC	- Main menu & other things.
RECMENU2.RSC	- Recipe naming menu.
RECMENU3.RSC	- Recipe coder menu.
RECEDMENU1.RSC	- Main editing menu.
RECEDMENU2.RSC	- Recipe duplicating menu.
RECEDMENU3.RSC	- Line by line editing menu.
RECEDMENU4.RSC	- Recipe renaming menu.
RECEDMENU5.RSC	- Ingredient changing menu.
RECEDMENU6.RSC	- Fat/Water/Yield changing menu.
RECANA.RSC	- Analysis menu.
RECMMS.RSC	- Food group selection menu.

HELP FILES

RECMENU1.HLP	- Help for main menu & other things.
RECMENU2.HLP	- Recipe naming help.
RECMENU3.HLP	- Recipe coder help.
RECEDMENU1.HLP	- Main editing help.
RECEDMENU2.HLP	- Recipe duplicating help.
RECEDMENU3.HLP	- Line by line editing help.
RECEDMENU4.HLP	- Recipe renaming help.

RECEDMENU5.HLP	- Ingredient changing help.
RECEDMENU6.HLP	- Fat/Water/Yield changing help.
RECANA.HLP	- Analysis help.
RECMMS.HLP	- Food group selection help.

MISCELLANEOUS

MMS.DAT	- Food groups used in menu selection.
RECIPE.FIL	- Contains names of database files.
RECIPE.OPT	- Project link file.

3.6.3 Subroutines & Functions

The following section shows each of the subroutines and functions in each of the source files for the Recipe CEA:

RECIPE.FOR - Main menu, coding functions, etc.

MAKE_RECIPE_FILE	- Main routine (main menu, etc.)
DORECIPE	- Recipe naming menu.
PROCESSRECIPE	- Recipe coding options menu.
POSTMESSAGE	- Put message in status box.
DOINGREDIENT	- Let user choose ingredient.
DELETRECORD	- Delete last user entry.
SUB-RECIPE	- Open or close a sub-recipe.
PROCESSCHANGES	- Fat/water/yield changes.
RECIPEDONE	- Complete current recipe.
INITSTATBOX	- Put up status box.
UPDATELEVEL	- Update sub-rec num in status box.
UPDATELINE	- Update line number in status box.
INITMAINRES	- Get main menu and other resources.
OPENRECFILE	- Open recipe file (new or old).
GETFOODGROUP	- Food group selection menu.
CHOICE	- Handle a numbered list of choices.
GETPCODE	- Get processing codes (retentions).
GETITYPE	- Get type of fat/water/yield change.
GETFAT	- Get ingredient for fat source.
ISCAN	- Find matching ingreds in database.
MODIFIERS	- Get fat/water/yield change info.
EOS	- Find end of non-NULL-term string.
FILE_OUT	- Write out coded recipe scratch file.
MISSING	- Log missing ingredients.
STARTUP	- Read in ingredients & other init.
ASK	- Ask user for input.
CHECK	- Translate coded recipe file.

READREAL	- Input a real number.

<u>INGRED.FOR</u> - Prompt for ingredients, units, etc.

GETINGREDIENT	- Ask user for ingredient name.
GETUNIT	- Ask user for unit of measure.
GETMULT	- Ask user for quantity of ingredient.
USCAN	- Handle units choices.

<u>RECEDIT.FOR</u> - Recipe editing functions.

RECEDIT	- Main editing menu.
GETRECIPECODE	- Ask user for recipe code to edit.
DELETERECIPE	- Delete current recipe from file.
DUPLICATERECIPE	- Duplicate current recipe.
INSERTDUPLICATE	- Insert duplicate recipe into file.
READINRECIPE	- Read recipe from file into memory.
WRITEOUTRECIPE	- Write edited recipe out to file.
EDITLINES	- Line by line editing menu.
NEXTPREVLINE	- Advance/backup in recipe one line.
CLEARMESSAGE	- Clear status message in box.
GOLINE	- Go directly to a recipe line.
GETLINETYPE	- Get recipe line type.
DELETELINE	- Delete current line from recipe.
INSERTINGRED	- Insert an ingredient line.
INSERTMODIFIER	- Insert fat/water/yield change line.
RECSUBSET	- Create recipe subset file.
OPENSUBSET	- Create/open recipe subset file.
EXTRACTRECIPE	- Extract a recipe for subset file.
BATCH	- Check for ".BAT" file.
BATCHRECIPE	- Parse batch file for subset/combine.
CHANGELINE	- Check which line type to edit.
CHANGEHEADER	- Edit the recipe header.
CHANGEINGRED	- Menu for editing ingredient line.
CHANGEWHOLEINGRED	- Edit whole ingredient line.
CHANGEUNIT	- Edit the unit of measure.
CHANGEAMOUNT	- Edit amount of ingredient.
CHANGEPROCS	- Edit process codes of ingredient.
CHANGEYIELD	- Menu for editing fat/water/yield.
CHANGEMODS	- Edit fat/water/yield or amount.
CHANGEFAT	- Edit fat source for current line.
COMBINERECFILES	- Get file names for combining.
COMBINERECIPE	- Combine a recipe with file.

RECIPE_ANA.FOR	- Main analysis routines.
RECIPE_ANA	- Init analysis and read recipe lines.
READ_RECLINE	- Read a line of the recipe.
SET_UP.FOR	- Analysis menu and initialization.
SET_UP	- Init and call analysis menu.
ANALYSISMENU	- Handle analysis menu.
COPYSTRING	- Copy string, pad w/ blanks & NULL.
MENUSTAT	- Put status message at menu bottom.
READ_OLDDEFS	- Read old parameters for menu.
ASSIGN_LABELS	- Assign nutrient labels to array.
CONVERT_NUTNUMS	- Parse nutrient number string.
HELP_NUTS	- Show nutrient label help.
READ_CHANGE	- (Not used anymore.)
MAKE_CHANGE	- (Not used anymore.)
GET_FILNAM	- (Not used anymore.)
CHECK_REP	- (Not used anymore.)
TOGGLE	- (Not used anymore.)
GET_NUTLIST	- (Not used anymore.)
WRITE_NEWDEFS	- Write out new parameters to file.
OPEN_FILES	- Open all files needed in analysis.
SET_INIT	- Calculate report parameters.
GET_YIELDS.FOR	- Parse fat/water/yield changes in analysis.
GET_YIELDS	- Parse fat/water/yield changes line.
REPORT.FOR	- Output analysis report.
REPORT	- Generate report file.
MAKE_FORMAT	- Create format string for report.
INGRED_ANA.FOR	- Ingredient analysis & retentions.
INGREDIENT	- Process ingredient line.
CVT_NUM	- Converts a string to real.
ADJUST_RETS	- Adjust for process codes (retentions).

NEW_RECIPE.FOR	- Handle first line of a recipe.
NEW_RECIPE	- Handle first line of a recipe.
PROCESS_DEL.FOR	- Sub-recipe closing, adjust yields, etc.
PROCESS_DEL	- Finish pending sub-recipe analysis.
ADJUST_YIELDS	- Adjust for fat/water/yield changes.
PERC_ADJUST	- Adjust for simultaneous fat/water.

3.7 Using the Oracle Data Loader for the Recipe Coder / Editor / Analyzer

3.7.1 Introduction

The Recipe CEA is an interactive program on the VAX for creating and editing coded recipe files and then analyzing them for nutritional composition. When the analysis is complete, the analyzed data is put into an Oracle table for later interfacing it with the data collected in the field. This document explains how to create an Oracle table for maintaining the data and how to put the analyzed output from the Recipe CEA into the table.

3.7.2 Creating an Oracle Table

The format of the table to be created for maintaining the analyzed output is exactly the same as the format for the ingredients database. In other words, each recipe can be viewed as if it were an ingredient with various nutrients. Notice that by keeping the same table format, the Dumptable program may be used for translating the Oracle table back into a Fortran format. For exact Oracle table formats, see the Oracle Table Translator document.

To create a new Oracle table, you could log into Oracle and do it interactively. However, since this table format has so many fields, it is better to prepare an SQL command file in advance and then run it in Oracle. The following shows the format of the SQL file needed to create a table with all of the nutrient fields:

```
CREATE TABLE TESTRECIPES
        (CODE   CHAR(10),
        NAME    CHAR(80),
        DESC1   CHAR(20),
        DESC2   CHAR(20),
        DESC3   CHAR(20),
        DESC4   CHAR(20),
        DESC5   CHAR(20),
        DESC6   CHAR(20),
MAJ             NUMBER(1),
MIN             NUMBER(1),
SUB             NUMBER(1),
A               NUMBER(1),
B               NUMBER(1),
T               NUMBER(1),
MRE             NUMBER(1),
ING             NUMBER(1),
H2O             NUMBER,
CAL             NUMBER,
PROT            NUMBER,
FAT             NUMBER,
CHO             NUMBER,
SUC             NUMBER,
SUG             NUMBER,
COM             NUMBER,
CRF             NUMBER,
DFIB            NUMBER,
DFI             NUMBER,
DFS             NUMBER,
ASH             NUMBER,
C               NUMBER,
THI             NUMBER,
RIB             NUMBER,
NIA             NUMBER,
B6              NUMBER,
FOL             NUMBER,
B12             NUMBER,
BIO             NUMBER,
PAN             NUMBER,
ARE             NUMBER,
AIU             NUMBER,
CARRE           NUMBER,
CAR             NUMBER,
D               NUMBER,
ETOT            NUMBER,
TTOC            NUMBER,
TOC             NUMBER,
VITK            NUMBER,
CA              NUMBER,
P               NUMBER,
MG              NUMBER,
FE              NUMBER,
FEH             NUMBER,
FEN             NUMBER,
ZN              NUMBER,
I               NUMBER,
CU              NUMBER,
MN              NUMBER,
FL              NUMBER,
CR              NUMBER,
SE              NUMBER,
MO              NUMBER,
K               NUMBER,
NA              NUMBER,
CL              NUMBER,
S               NUMBER,
AL              NUMBER,
BA              NUMBER,
BO              NUMBER,
SR              NUMBER,
SI              NUMBER,
VA              NUMBER,
CO              NUMBER,
NI              NUMBER,
ARS             NUMBER,
TIN             NUMBER,
SAT             NUMBER,
S4              NUMBER,
S6              NUMBER,
S8              NUMBER,
S10             NUMBER,
S12             NUMBER,
S14             NUMBER,
S16             NUMBER,
S18             NUMBER,
MON             NUMBER,
M16             NUMBER,
M18             NUMBER,
M20             NUMBER,
M22             NUMBER,
POL             NUMBER,
P182            NUMBER,
P183            NUMBER,
P184            NUMBER,
P204            NUMBER,
P205            NUMBER,
P225            NUMBER,
P226            NUMBER,
CHOL            NUMBER,
PHY             NUMBER,
HIS             NUMBER,
ILE             NUMBER,
LEU             NUMBER,
LYS             NUMBER,
MET             NUMBER,
PHE             NUMBER,
THR             NUMBER,
TRY             NUMBER,
VAL             NUMBER,
ALA             NUMBER,
ARG             NUMBER,
ASP             NUMBER,
CYS             NUMBER,
GLU             NUMBER,
GLY             NUMBER,
PRO             NUMBER,
SER             NUMBER,
TYR             NUMBER,
MC              NUMBER,
TP              NUMBER,
CAF             NUMBER,
ETH             NUMBER,
CARN            NUMBER,
INO             NUMBER,
PHT             NUMBER,
CHL             NUMBER);
```

In this example, we are creating a table called TESTRECIPES. The SQL file will be called TEST.SQL. To run this file, first log into Oracle by typing "SQL". When the prompt appears, type "START TEST" to execute the file TEST.SQL. After the table is created, you can verify the fields by typing "DESCRIBE TESTRECIPES". This will show all of the field names and types.

3.7.3 Dumping Analyzed Output into the Oracle Table

Once the Oracle table has been created, and the recipes have been analyzed with the Recipe CEA, the analyzed output file can be dumped into the Oracle table. (Note: the analyzed output file is not the same as the report file. The report file usually ends with ".REP" and the analyzed output file usually ends with ".ANA".)

The data can be dumped with the Oracle Data Loader (ODL) program. To use this program, you must first prepare an ODL command file. This file tells ODL the format of the dumptable Fortran file (the analyzed output), and the Oracle table (created in the previous section). The following shows the format of the ODL file needed to dump the analyzed output into an Oracle file:

```
DEFINE RECORD REC1 AS
       FLD1 (CHAR(10)),
       FLD2 (CHAR(80)),
       FLD3 (CHAR(20)),
       FLD4 (CHAR(20)),
       FLD5 (CHAR(20)),
       FLD6 (CHAR(20)),
       FLD7 (CHAR(20)),
       FLD8 (CHAR(20)),
       FLD9 (CHAR(1)),
       FLD10 (CHAR(1)),
       FLD11 (CHAR(1)),
       FLD12 (CHAR(1)),
       FLD13 (CHAR(1)),
       FLD14 (CHAR(1)),
       FLD15 (CHAR(1)),
       FLD16 (CHAR(1)),
       NUT1  (CHAR(13)),
       NUT2  (CHAR(13)),
       NUT3  (CHAR(13)),
       NUT4  (CHAR(13)),
       NUT5  (CHAR(13)),
       NUT6  (CHAR(13)),
       NUT7  (CHAR(13)),
       NUT8  (CHAR(13)),
       NUT9  (CHAR(13)),
       NUT10 (CHAR(13)),
       NUT11 (CHAR(13)),
       NUT12 (CHAR(13)),
       NUT13 (CHAR(13)),
       NUT14 (CHAR(13)),
       NUT15 (CHAR(13)),
       NUT16 (CHAR(13)),
       NUT17 (CHAR(13)),
       NUT18 (CHAR(13)),
       NUT19 (CHAR(13)),
       NUT20 (CHAR(13)),
       NUT21 (CHAR(13)),
       NUT22 (CHAR(13)),
       NUT23 (CHAR(13)),
       NUT24 (CHAR(13)),
       NUT25 (CHAR(13)),
       NUT26 (CHAR(13)),
       NUT27 (CHAR(13)),
       NUT28 (CHAR(13)),
       NUT29 (CHAR(13)),
       NUT30 (CHAR(13)),
       NUT31 (CHAR(13)),
       NUT32 (CHAR(13)),
       NUT33 (CHAR(13)),
       NUT34 (CHAR(13)),
       NUT35 (CHAR(13)),
       NUT36 (CHAR(13)),
       NUT37 (CHAR(13)),
       NUT38 (CHAR(13)),
       NUT39 (CHAR(13)),
       NUT40 (CHAR(13)),
       NUT41 (CHAR(13)),
       NUT42 (CHAR(13)),
        NUT43  (CHAR(13)),
        NUT44  (CHAR(13)),
        NUT45  (CHAR(13)),
        NUT46  (CHAR(13)),
        NUT47  (CHAR(13)),
        NUT48  (CHAR(13)),
        NUT49  (CHAR(13)),
        NUT50  (CHAR(13)),
        NUT51  (CHAR(13)),
        NUT52  (CHAR(13)),
        NUT53  (CHAR(13)),
        NUT54  (CHAR(13)),
        NUT55  (CHAR(13)),
        NUT56  (CHAR(13)),
        NUT57  (CHAR(13)),
        NUT58  (CHAR(13)),
        NUT59  (CHAR(13)),
        NUT60  (CHAR(13)),
        NUT61  (CHAR(13)),
        NUT62  (CHAR(13)),
        NUT63  (CHAR(13)),
        NUT64  (CHAR(13)),
        NUT65  (CHAR(13)),
        NUT66  (CHAR(13)),
        NUT67  (CHAR(13)),
        NUT68  (CHAR(13)),
        NUT69  (CHAR(13)),
        NUT70  (CHAR(13)),
        NUT71  (CHAR(13)),
        NUT72  (CHAR(13)),
        NUT73  (CHAR(13)),
        NUT74  (CHAR(13)),
        NUT75  (CHAR(13)),
        NUT76  (CHAR(13)),
        NUT77  (CHAR(13)),
        NUT78  (CHAR(13)),
        NUT79  (CHAR(13)),
        NUT80  (CHAR(13)),
        NUT81  (CHAR(13)),
        NUT82  (CHAR(13)),
        NUT83  (CHAR(13)),
        NUT84  (CHAR(13)),
        NUT85  (CHAR(13)),
        NUT86  (CHAR(13)),
        NUT87  (CHAR(13)),
        NUT88  (CHAR(13)),
        NUT89  (CHAR(13)),
        NUT90  (CHAR(13)),
        NUT91  (CHAR(13)),
        NUT92  (CHAR(13)),
        NUT93  (CHAR(13)),
        NUT94  (CHAR(13)),
        NUT95  (CHAR(13)),
        NUT96  (CHAR(13)),
        NUT97  (CHAR(13)),
        NUT98  (CHAR(13)),
        NUT99  (CHAR(13)),
        NUT100  (CHAR(13)),
        NUT101  (CHAR(13)),
        NUT102  (CHAR(13)),
        NUT103  (CHAR(13)),
        NUT104  (CHAR(13)),
        NUT105  (CHAR(13)),
        NUT106  (CHAR(13)),
        NUT107  (CHAR(13)),
        NUT108  (CHAR(13)),
        NUT109  (CHAR(13));
DEFINE  SOURCE FILE
        FROM [NUTRITION.RECIPE]TESTMODS.ANA
        LENGTH  1635
        CONTAINING REC1;
```

```
FOR EACH RECORD
        INSERT INTO TESTRECIPES
        (CODE,NAME,DESC1,DESC2,DESC3,DESC4,DESC5,DESC6,
        MAJ,MIN,SUB,A,B,T,MRE,ING,
        H2O,CAL,PROT,FAT,CHO,SUC,SUG,COM,CRF,DFIB,
        DFI,DFS,ASH,C,THI,RIB,NIA,B6,FOL,B12,BIO,
        PAN,ARE,AIU,CARRE,CAR,D,ETOT,TTOC,TOC,VITK,
        CA,P,MG,FE,FEH,FEN,ZN,I,CU,MN,FL,CR,SE,
        MO,K,NA,CL,S,AL,BA,BO,SR,SI,VA,CO,NI,ARS,
        TIN,SAT,S4,S6,S8,S10,S12,S14,S16,S18,
        MON,M16,M18,M20,M22,POL,P182,P183,P184,
        P204,P205,P225,P226,CHOL,PHY,HIS,ILE,
        LEU,LYS,MET,PHE,THR,TRY,VAL,ALA,ARG,ASP,
        CYS,GLU,GLY,PRO,SER,TYR,MC,TP,CAF,ETH,
        CARN,INO,PHT,CHL) VALUES
        (FLD1,FLD2,FLD3,FLD4,FLD5,FLD6,FLD7,FLD8,FLD9,FLD10,
        FLD11,FLD12,FLD13,FLD14,FLD15,FLD16,
        NUT1,NUT2,NUT3,NUT4,NUT5,NUT6,NUT7,NUT8,NUT9,
        NUT10,NUT11,NUT12,NUT13,NUT14,NUT15,NUT16,
        NUT17,NUT18,NUT19,NUT20,NUT21,NUT22,NUT23,
        NUT24,NUT25,NUT26,NUT27,NUT28,NUT29,NUT30,
        NUT31,NUT32,NUT33,NUT34,NUT35,NUT36,NUT37,
        NUT38,NUT39,NUT40,NUT41,NUT42,NUT43,NUT44,
        NUT45,NUT46,NUT47,NUT48,NUT49,NUT50,NUT51,
        NUT52,NUT53,NUT54,NUT55,NUT56,NUT57,NUT58,
        NUT59,NUT60,NUT61,NUT62,NUT63,NUT64,NUT65,
        NUT66,NUT67,NUT68,NUT69,NUT70,NUT71,NUT72,
        NUT73,NUT74,NUT75,NUT76,NUT77,NUT78,NUT79,
        NUT80,NUT81,NUT82,NUT83,NUT84,NUT85,NUT86,
        NUT87,NUT88,NUT89,NUT90,NUT91,NUT92,NUT93,
        NUT94,NUT95,NUT96,NUT97,NUT98,NUT99,NUT100,
        NUT101,NUT102,NUT103,NUT104,NUT105,NUT106,
        NUT107,NUT108,NUT109)
NEXT RECORD
```

In this example, the ODL file will be called "TEST.ODL" and the analyzed recipe file is called "TESTMODS.ANA". To run this file through the Oracle Data Loader, type the following command at the DCL prompt:

```
$ ODL TEST.ODL TEST.LOG NUTRITION/NUT
```

The "TEST.LOG" file will contain errors and results when complete. The "NUTRITION/NUT" is the Oracle account and password for Nutrition.

The data are now loaded into an Oracle table. Dumptable can be used on this table to create a FORTRAN keyed indexed file for use as input ingredient data for other nutrition programs (i.e. Diet History Analyzer).

CHAPTER 4.0

NUTRITION DATA ENTRY PROGRAMS

Contributing Authors:
John Finn
William Wisnaskas
Carlo Radovsky
Dr. Kenneth Samonds

4.1 Food Frequency Questionnaire

Objective:

A research instrument to be used in Army dietary studies to evaluate prior eating patterns and nutrient intakes.

Major Points:

1. Food list based on Gladys Bloch's food list from analysis of the Nationwide Food Consumption Survey.
    a. Includes major contributors to nutrient intakes.
    b. Devised to measure ~12 nutrients (including sodium and cholesterol).
    c. Included foods are also major sources for ~8 other nutrients.
    d. Includes approximately 100 foods.
    e. Bloch's method will account for 80-95% of the total nutrient intake for a group of people.
    f. Advantage is that it measures food intake over a longer period of time compared to diet histories or records.
    g. Disadvantage is that it depends on memory and portion estimation by the subject.
2. Food list will be designed for use on an optically scanned form.
    a. Data gathered on frequency of consumption per day, week, or month over the past year.
    b. Options of small medium, or large servings which will be age and sex specific.
3. Will generate a coded data file which will be analyzed with USARIEM's standard dietary intake program (ANALYZE).
    a. Determine average daily intakes of these 20 nutrients and food group consumption patterns.
    b. Able to compare to age and sex specific RDAs or MRDAs.

Future Plans

Project expected to be completed by December 1989.

4.2 Food Consumption Data Entry User's Guide

4.2.1 Introduction

The Food Consumption Data Entry, or FOODDE, is an interactive program on the VAX and PC for entering food consumption data. This document describes how to use the FOODDE program. Currently, this document should be considered a "living document".

Food consumption by individuals is manually recorded in the field. Prior to the study, a COD file is created that contains a listing of a food item or recipe contained in the study table, the unit of measure, the gram weight, a food classification code, and a three digit code to be associated with this item. This program allows an individual's food consumption of items listed in the COD file to be entered and stored in files so that it may later be analyzed.

4.2.2 Getting Started

To run the FOODDE program several files are needed. These files must be contained in the same subdirectory on the VAX or PC, except for the executable file, which may be left in the main VAX nutrition or PC directory. The following is a listing of all the files:

Program files:	File Name Description:
FOODDE.EXE	The main executable program
FOOD.DEF	File of default values
*.COD	A CODE file of food items
*.TPL	A default template file

Copy the FOOD.DEF, *.COD, and the *.TPL files into the directory from where you will be running the program.

To start the program, type at the DCL or PC prompt:

```
FOODDE
```

This will invoke the FOODDE program. At this time, the file FOOD.DEF will be accessed to find the default values that had been previously stored. If this file is not found, an error message will display. See section 4.2.3.5 for the proper procedure to follow in case of this event.

4.2.3 The Main Menu

The main menu (Figure 4.1) currently contains 6 items. Main menu options are selected by entering a number 1-6. Any other numeric entry, or any non-numeric entry at the main menu will cause an error message to display. Only option 6 on the main menu will allow you to exit from the FOODDE program.

FIGURE 4.1 Food Data Entry Main Menu

```
FOOD CONSUMPTION Program

 (1) Enter Food Data
 (2) Edit Food Data
 (3) Produce a Report
 (4) Create/Print Template
 (5) Default Values
 (6) Exit Program


Please choose option 1-6:
```

4.2.3.1 Enter Food Data

After "Enter Food data" is selected from the main menu by entering a "1", the names of the default COD file being used and the default TEMPLATE file being used will display (refer to Appendix C for more information about the COD file and Appendix D for more information on the TEMPLATE file).

```
USING DEFAULT COD FILE: codfile.cod
USING TEMPLATE FILE : templatefile.ext
```

These values are contained in the FOOD.DEF defaults file and may only be changed in the "Default Values" option (section 4.2.3.5); It is suggested you read this section before entering or editing data so that the significance of these defaults is not overlooked. Also, refer to section 4.2.3.4 for information on TEMPLATE files.

The following prompt will then appear:

FIGURE 4.2 ENTER DATA FILE NAME :

You are encouraged to enter full file names (filename.ext). The file name specified here will now house the data that shall be entered. Entering a <CR> for the data file name will bring you back to the Main Menu at this time. If the name entered refers to a file that exits and contains data, an error message will display and Figure 4.2 will prompt you for another name.

The TEMPLATE file will now display to the screen. All the active fields shall be shown in their previously specified positions that are contained in the TEMPLATE file. You will then be prompted to enter values for all the fields displayed according to the order specified by the TEMPLATE. The value entered will be immediately displayed to the screen in its proper position.

Certain fields have information concerning the translation between the numeric entry and its actual value outside of the program that can be viewed by entering a "?" for the value of the field. See Appendix B for the fields that have this feature and their translations.

When you have finished entering a record, the following prompt will appear:

FIGURE 4.3 ENTER MORE RECORDS <CR>, 1 TO QUIT

Enter a "1" at this time to indicate that you have entered all the data for this file and you wish to save it. The data will then be written to the data file and the Main Menu will appear.

Press <CR> to continue entering data records. The following prompt will then be displayed:

FIGURE 4.4 USE DEFAULT VALUES <CR>, 1 FOR NO:

Default values refer to special fields which may obtain values from the previous record's value. This eliminates having to enter the same value for the same field repetitively. You can make a field a default field when creating a template which will be discussed in a later section. Those fields specified as default fields will display their default values, so you can verify that you want to use them. Press <CR> to use them. You will then be prompted to enter a value for the next non-default field in the TEMPLATE.

Enter a "1" to indicate that you do not want to use the default values. You will then enter values for every field of the record.

If you decide that you do not wish a record to be written to the data file, enter "\Q" at the field prompt. The message "PROCESS TERMINATED BY USER REQUEST" will display. This will tell the FOODDE program that you wish to "quit" out of this record without saving it. Figure 4.3 will then be displayed.

To aid the data enterer, the last record's food item will be displayed at the bottom of the template after the string "LAST ITEM ENTERED :" to prevent the entering of duplicate records accidentally.

As you enter records, the FOODDE program is internally organizing them according to the SUBJECT NUMBER field and the DATE field, if they are active fields. The resulting data file will therefore be in ascending order by subject number, and records with the same subject number will be in ascending order by date. If the SUBJECT NUMBER and/or the DATE field(s) is(are) not active, the data file will be listed in the order that the records are entered.

4.2.3.2 Edit Food Data

Entering a "2" at the Main Menu will allow you to edit an existing data file. The default COD file and TEMPLATE file name will display to the screen, as they would if you selected to enter data. Figure 4.2 will be displayed, and you must enter the name of an existing data file to edit. If the data file name entered does not exist, an error message will display, and Figure 4.2 will be displayed again.

Once you have created and stored a data file using "Enter Food Data", you may only access this file by using the Edit feature. For example, if subsequent data are collected for the remainder of the study, this data will have to be entered using the Edit feature or by creating another file and later merging the files.

The TEMPLATE file will display at this time. It is important to realize that a data file is NOT internally linked to any specific TEMPLATE file. Thus, data entered using one TEMPLATE may be edited using another. However, this is not recommended due to the fact that TEMPLATES can vary greatly in the number of active fields, their positions, and their ranges. It is important to connect a data file with a specific TEMPLATE file, unless a deliberate change is desired.

The following options are available when editing a data file:

FIGURE 4.5

```
(1)CHANGE RECORD    (2)ADD RECORD    (3)DELETE RECORD
        SELECT OPTION (1-3), <CR> TO QUIT :
```

Editing is comprised of three major actions: changing an existing record, adding new records, and deleting an entire existing record. You must choose which action you wish to perform at this time. Pressing <CR> will bring you back to the Main Menu.

<u>CHANGE RECORD</u>

Entering a "1" to select CHANGE RECORD will cause the following to be displayed:

FIGURE 4.6

```
ENTER RECORD NUMBER, <CR> TO QUIT
```

The record number is a unique sequential number associated with every record of the data file. It is generated when the file is read into memory by the program. You may see a record's associated record number when you produce a report from a data file (section 4.2.3.3). Enter the number of the record that you wish to change. Only numeric entries between 1 and the number of records in the file are valid. A <CR> at this time will bring you back to the Select Edit Options Menu

(Figure 4.5).

The record specified will display to the screen according to the TEMPLATE format. The fields may now be changed by choosing the number associated with the field. The following prompt will be displayed:

FIGURE 4.7 ENTER FIELD NUMBER, <CR> TO QUIT:

Enter the number of the field you wish to change, and the field prompt associated with it will display, asking you for the new value. You may change as many fields as you like and as many times as you like. The fields that have translation information will display it if the "?" is entered at this time (see Appendix B).

When you are finished with the present record or if you do not wish to change the record at all, press <CR>. This will bring you back to Figure 4.6 (Enter Record Number.

If the SUBJECT NUMBER and/or DATE field(s) are active for this data file and you change their values, the position of the record in the data file may need to be updated so that the data file will always be in ascending order by subject number, and within subject number, by date. This would result in a line number change, so you may wish to reproduce a report of this data file to assist you with the editing.

ADD RECORD

Entering a "2" for ADD RECORD will cause the TEMPLATE to appear. This option operates basically the same as entering a record using ENTER FOOD DATA. The default values and last item entered will display, you have the option to quit from a record, and so on.

The new record is inserted in the data file according to the SUBJECT NUMBER and the DATE field, conforming to the correct data file order as described in section 4.2.3.1.

After adding a new record, Figure 4.5 will appear, giving you the opportunity to stop entering new records.

DELETE RECORD

Entering a "3" for DELETE RECORD will cause Figure 4.6 to be displayed. Enter the number of the record that you wish to delete at this time. The record will be displayed so that you can verify that it is the one that you wish to delete. The prompt for Figure 4.8 will be displayed:

FIGURE 4.8 DELETE THIS RECORD [Y/N]

The "N" or "NO" response is the default for this prompt. You must enter a "Y" in order to delete this record. Either response will bring you back to Figure 4.6.

If you have numerous records to delete, note that the record numbers are NOT updated until you exit the EDIT menu entirely. This was done so that after one record is deleted, the next record to be deleted can still be referenced by its original number, thus eliminating the need for mental mathematics to calculate its new record number.

4.2.3.3 Produce a Report

Entering a "3" at the Main Menu will allow you to produce a report of a data file. The default COD file and TEMPLATE file will display (See section 4.2.3.1) and Figure 4.2 will be displayed asking you to enter the data file name. Enter the name of the data file for which you wish to produce a report on, or enter <CR> to bring you back to the Main Menu. The data file specified must exist, or an error message will display. Once the data file name is entered, the following will be displayed:

FIGURE 4.9 ENTER THE REPORT FILE NAME:

You may name your report file any valid name you desire(ie. filename.ext). The report will NOT be automatically printed out. You must exit the FOODDE program in order to print out the file at the DCL or PC prompt.

NOTE: The Produce a Report function will generate a report according to the active TEMPLATE file. Only the fields that are active within this TEMPLATE will be in the report, and they will be ordered according to the TEMPLATE field positions. Using a different TEMPLATE than the one that the data file was created with to produce a report can be useful, but please remember, field values will only be correct when using a TEMPLATE that is a SUBSET of the original TEMPLATE file. This insures that the fields in the report were active originally, and therefore do not contain garbage values.

A sample report generated using a TEMPLATE file in which every field is active is included on the following page.

4.2.3.4 Create/Print Template

Entering a "4" at the Main Menu will create or print a TEMPLATE file. A TEMPLATE file serves as an interface between the user and the FOODDE program when the user is entering food data. By using a TEMPLATE, the user is able to enter data in different formats, but the data will be stored in the same format for

```
     File Used: SAMPLE.DAT
                                          Date: 14-SEP-1989       Time: 15:01:56.88

Study Name:  SAMPLE

                   Study Cycle                                      Amount Amount Amount                Salt   Data
Rnum  Date   Day  Day    Day  Group SubNum Sex Meal Time Source Taken  Ret    Cons   Reason Rating  Added Coll
---- ------ --- ----- ----- ----- ------ --- ---- ---- ------ ------ ------ ------ ------ ------ ------ ----
   1 890909  S     1     1     1      1    M    3 1700      1   2.00   0.00   2.00     25      8   0.00    1
         FoodNum: RL4306    Foodname: BEEF JERKY  JAN 88                                Unit name: ONE

   2 890909  S     1     1     1      1    M    3 1700      1   1.00   0.00   1.00     25      3   0.00    1
         FoodNum: RL4004    Foodname: BEVERAGE BAR- TROPICAL PUNCH  JAN 88              Unit name: ONE

   3 890909  S     1     1     1      1    M    2 1200      1   1.00   0.00   1.00     25      2   0.00    1
        FoodNum: RL4800    Foodname: DESERT BAR- APPLE CINN  JAN 88                    Unit name: ONE

   4 890909  S     1     1     1      1    M    2 1200      1   1.00   0.00   1.00     25      4   0.00    1
         FoodNum: RL4003    Foodname: BEVERAGE BAR- ORANGE  JAN 88                      Unit name: ONE

   5 890909  S     1     1     1      1    M    2 1200      1   1.00   0.00   1.00     25      7   0.00    1
         FoodNum: MR6816    Foodname: GUM - VALUES FROM PG. 168 B & CHURCH             Unit name: ONE

   6 890909  S     1     1     1      1    M    2 1200      1   1.00   0.00   1.00     25      4   0.00    1
         FoodNum: MRI911    Foodname: TABASCO SAUCE                                     Unit name: ONE

   7 890909  S     1     1     1      1    M    1 0900      1   1.50   0.00   1.50     25      7   0.00    1
         FoodNum: RL4204    Foodname: DAIRY BAR- STRAWBERRY  JAN 88                     Unit name: ONE

   8 890909  S     1     1     1      1    M    1 0900      1   1.00   0.00   1.00     25      6   0.00    1
         FoodNum: RL4406    Foodname: CEREAL BAR-BRAN FLAKE JAN 88                      Unit name: ONE
```

compatibility between studies.

The TEMPLATE is a subset of all the possible FOODDE data fields (Appendix B). The TEMPLATE is needed in order to enter data, edit data, and to produce a report. The TEMPLATE file name should relate to the study for which it is being used.

The following prompt will appear:

```
FIGURE 4.10     (1)CREATE TEMPLATE FILE   (2)PRINT TEMPLATE FILE
                 SELECT OPTION (1-2), <CR> TO QUIT:
```

Press <CR> at this time if you wish to return to the Main Menu.

CREATE TEMPLATE FILE

Enter a "1" to create a TEMPLATE file. You will be asked in order of the internal field sequence (Appendix B) if you wish to use the field or not.

```
FIGURE 4.11     USE fieldname [Y/N] ?
```

Enter a "Y" for yes to indicate that this field will be active in the template, or enter a "N" for no if you do not wish it to be active.

If you wish to terminate the creation of the template, enter "\Q" . The message "PROCESS TERMINATED BY USER REQUEST" will display, and this will bring you back to Figure 4.10. This option is only available to you when Figure 4.11 displays.

If you choose to make the field inactive, the next field name will appear, and Figure 4.11 will be displayed. Otherwise, the next prompt will be displayed:

```
FIGURE 4.12     ENTER THE POSITION FOR THE FIELD :
```

You may have any field positioned in any of the available 19 positions. Once the position is entered, the field name will display on the screen at that location. You may not overwrite a position you have previously specified, or an error message will be displayed.

```
FIGURE 4.13     MAKE FIELD A DEFAULT FIELD [Y/N]?
```

Making a field a default field means that when you are entering or adding records to a data file, this field will be displayed with the value you last entered for it, therefore giving you the option to use this value instead of re-entering a new one. Enter a "Y" for yes to make the field a default field, or a "N" for no, indicating that you wish to enter a value for this field for each record.

FIGURE 4.14 SPECIFY A RANGE FOR THE FIELD [Y/N]?

You are able to specify a range for your TEMPLATE fields. Enter a "Y" for yes if you want this field to have a range or a "N" for no range.

A "N" response will cause Figure 4.11 to display with the next field name. If a "Y" is entered, you will now specify the ranges desired.

FIGURE 4.15 ENTER THE LOW VALUE FOR THE FIELD :

FIGURE 4.16 ENTER THE HIGH VALUE FOR THE FIELD :

The field will have a low and high range specified by you. The valid high and low values are limited by the FOODDE field type (see Appendix B). An error will occur if you specify a high range that is smaller than the low range in value.

When you have been through all the fields, you will then be prompted for the TEMPLATE file name:

FIGURE 4.17 ENTER TEMPLATE FILE NAME :

Although not required, the extension .TPL is suggested for the TEMPLATE file name. The file may now be used as the new default TEMPLATE (section 4.2.3.5).

PRINT TEMPLATE FILE

Enter a "2" to generate a print out of a TEMPLATE file. A hard copy of a TEMPLATE file is very useful, since it contains all the fields that are active in the order of their positions, whether or not it is a default field, and the ranges that were specified.

Figure 4.17 will be displayed, and after you enter the TEMPLATE file name, a message will be displayed indicating that the output file has been generated. The output file name is created for you; it is the TEMPLATE file name with the extension .OUT. The print file can not be printed from within the FOODDE program. You must exit FOODDE to print it. If the file specified does not exist or contains no data, an error message will be displayed, and Figure 4.17 will prompt again.

On the following page is an example of a template file, SAMPLE.TPL, that has been translated to the file SAMPLE.OUT and printed.

```
Template File Used: SAMPLE.TPL
Date: 14-SEP-1989        Time: 14:52:28.73


Position:  1        Variable Name: STUDY
--------->Default field
      ------No Range Specified------

Position:  2        Variable Name: DATE
--------->Default field
      ------No Range Specified------

Position:  3        Variable Name: DAY
--------->Default field
      ------No Range Specified------

Position:  4        Variable Name: STUDY DAY
--------->Default field
       Low Value :     1      High Value :     5

Position:  5        Variable Name: CYCLE
--------->Default field
       Low Value :     1      High Value :     3

Position:  6        Variable Name: GROUP
--------->Default field
       Low Value :     1      High Value :     8

Position:  7        Variable Name: SUBJECT
--------->Default field
       Low Value :     1      High Value :    16

Position:  8        Variable Name: SEX
--------->Default field
      ------No Range Specified------

Position:  9        Variable Name: MEAL
--------->must always be specified
       Low Value :     1      High Value :     5

Position: 10        Variable Name: TIME
--------->must always be specified
      ------No Range Specified------

Position: 11        Variable Name: SOURCE
--------->must always be specified
       Low Value :     1      High Value :    10

Position: 12        Variable Name: FOOD ITEM
--------->must always be specified
       Low Value :     1      High Value :  9999

Position: 13        Variable Name: AMOUNT TAKEN
--------->must always be specified
      ------No Range Specified------

Position: 14        Variable Name: AMOUNT RET
--------->must always be specified
      ------No Range Specified------

Position: 15        Variable Name: AMOUNT CONS
--------->must always be specified
      ------No Range Specified------

Position: 16        Variable Name: REASON
--------->must always be specified
       Low Value :     1      High Value :    20

Position: 17        Variable Name: RATING
--------->must always be specified
      ------No Range Specified------

Position: 18        Variable Name: SALT ADDED
--------->must always be specified
      ------No Range Specified------

Position: 19        Variable Name: COLLECTOR
--------->must always be specified
      ------No Range Specified------
```

4.2.3.5 Default Values

Enter a "5" at the Main Menu (fig 4.1) to change or view the default values that are stored in the unformatted external file FOOD.DEF. These are values that will probably not need to be changed, except between different studies. The following will be displayed:

FIGURE 4.18

```
D E F A U L T   V A L U E S

(1) COD FILE
(2) TEMPLATE FILE
(3) STUDY NAME
(4) GROUP DESCRIPTIONS
(5) DATA COLLECTOR NAMES

SELECT OPTION (1-5), <CR> TO QUIT:
```

Pressing <CR> at this time will bring you back to the Main Menu.

Enter a "1" to change the default COD file name. The default COD file name will be displayed, and then the prompt:

FIGURE 4.19 `ENTER COD FILE NAME:`

will be displayed. The specified COD file must exist, or an error message will display. Press <CR> at this time if you decide not to change the default COD file name.

Enter a "2" to change the default TEMPLATE file name. The default TEMPLATE file name will display with the following prompt:

FIGURE 4.20 `ENTER THE TEMPLATE FILE NAME:`

The TEMPLATE file must exist or an error message will display. If you press <CR> at this time, the default TEMPLATE file name will not be changed.

Enter a "3" to change the default study name. The default study name will display, and so will the prompt:

FIGURE 4.21 `ENTER THE STUDY NAME:`

The study name is an actual FOODDE field (see Appendix B) and the value entered here will be placed into any new data file which is created using a TEMPLATE in which the field STUDY NAME is active. Press <CR> at this time if you do not wish to change the study name.

Enter a "4" to change the group descriptions. These descriptors are determined by the design of the study. For example, the types of rations being eaten by different groups participating in a study (i.e., MRE, RCW, and RLW) may be the names of the different groups. The group descriptions can later be viewed to aid you in selecting a number when you are entering a value for the GROUP field. The present default group descriptions will display first, and then the following prompt will appear:

FIGURE 4.22 ENTER THE GROUP DESCRIPTIONS, <CR> TO QUIT:

Below this prompt numbers will display sequentially (max 30) and you may enter the new group descriptions at this time next to their appropriate number. Pressing <CR> at number 1 indicates you do not wish to change the group descriptions. When you finish entering all the new group descriptions, press <CR> to indicate that you are done at the next number prompt.

Enter a "5" to change the Data Collector names. The Data Collector names can later be viewed to aid you in selecting a data collector number when you are entering a value for the DATA COLLECTOR field. The default Data Collector names will display first, and then the following prompt will appear:

FIGURE 4.23 ENTER THE DATA COLLECTOR NAMES, <CR> TO QUIT:

Below this prompt numbers will display sequentially (max 30) and you may enter the new Data Collector names at this time next to their appropriate number. Pressing <CR> at number 1 indicates that you do not wish to change any of the Data Collector names. When you finish entering all the Data Collector names that are needed, press <CR> to indicate you are done at the next number prompt.

It is possible for the FOOD.DEF default file to be deleted or corrupted accidentally. This will cause an error message to appear at the main menu when the default file is being accessed. If this happens, immediately select DEFAULT VALUES from the Main Menu and enter in the values desired.

4.3 Food Consumption Data Entry Programmer's Guide

4.3.1 Introduction

The FOODDE program is an interactive program on the VAX and PC for entering food consumption data. This document describes some of the details on how the program works, and it is intended more for the programmer than the user.

4.3.2 Program Source Code Files

The FOODDE program resides in two Pascal files;

FOODDE.PAS	Main routines, main functions
ALLSUBS.PAS	Field, screen routines

The file ALLSUBS.PAS is handled as an include file by the FOODDE.PAS file. The VAX version of FOODDE is linked with the VAX SYSLIB at the time of compilation by typing:

```
$ LINK/SYSLIB FOODDE
```

With the PC version, this is not necessary (or possible!).

Although great care was taken to make the two versions of the FOODDE program compatible, there are some unavoidable differences between the programs. The major areas affected are; Screen management, Substring selection, File openi..g, and File existence verification.

These files should be located in the same directory as the executable, FOOD.EXE, or in a designated source code directory.

4.3.3 Subroutines and Functions

The source code files contain many subroutines and functions, and they are named below.

<u>FOODDE.PAS</u>

FREE_NODES	Free nodes (memory) allocated by the program.
GET_COD	Read code file into doubly linked list.
FILL_RECORD	Get user to enter data for a record.
WRITE_DATA	Write the food data to output file.
GET_TEMPLATE	Read in template file.
PRINT_FIELD	Print translation of template file.
TEMPLATE_MENU	Menu to create or print a template.
INSERT_IT	Inserts a data record into food linked list.
ENTER_DATA	Enter records to new file, add them to old file.
READDATA	Read into linked list an existing food data file.
DELETE_RE ORD	Delete a food data record from the linked list.
UNLINK_NC E	Unlink a data node from the food linked list.
CHANGE_REC	Perform changes to fields of a record.
EDIT_REC	Edit menu: changing, adding, deletion.
INIT_INFO	Initialize the arrays for the program at startup.
INSCHARS	Insert report output characters into output line.
CONVERT_REC	Convert all data fields to chars to print report.
CREATE_HEADING	Create heading for report.

REPORT	Generate a report from food data file.
DEFAULTS	Read in or write out the FOOD.DEF file.
DEF_VALS	Change/view defaults values from FOOD.DEF.

<u>ALLSUBS.PAS</u>

GET_DIGIT	Convert character digit into numerical equivalent.
CLEAR_IT	Clear screen from a line and column to the end.
PUT_STRING	Display a string at a line,column position.
PUT_CHARS	Put character data fields to screen.
PUT_INT	Put integer data fields to screen.
PUT_REAL	Put real data fields to screen.
FIELD_TO_SCREEN	Put a record field prompt to screen position.
SET_SCREENS	Put all the field prompts active to the screen.
ERR_MESS	Display an error message in proper area.
CHAR_TO_INT	Convert character string into integer.
CHAR_TO_REAL	Convert character string into real.
GET_INPUT	Get a user entered string from the screen.
GET_INT	Get a user entered integer from the screen.
GET_REAL	Get a user entered real from the screen.
HELP_LINE	Display help information about field.
HELP_INFO	Obtain GROUP and COLLECTOR array values.
DATA_FILE	Open a data file by HANDLE: 1=old, 0=new.
TEMPLATE_FILE	Open template file by HAndle: 1=old, 0=new.
COD_FILE	Open COD file.
DISPLAY_FOOD	Display food description of food item entered.
DISPLAY_REC	Place all record data values to the screen.
LOCATE_FOOD	Locate in the COD linked list the Food item.
PLACE_STUDY	Put the study name the screen.
GET_STUDY	Get the study name from the user.
CHECK_DATE	Verify that the date entered is valid.
GET_DATE	Get the date of consumption from user.
GET_SUBJECT	Get the subject number from the user.
GET_SEX	Get the subject's sex from user.
GET_GROUP	Get the subject's group number from user.
GET_CYCLE	Get the cycle day from the user.
GET_MEAL	Get the meal number from the user.
GET_TIME	Get the military time of consumption from user.
GET_FOODNUM	Get the food item number from user.
GET_AMT_TAKE	Get the amount taken value from the user.
GET_AMT_CONS	Get the amount consumed value from user.
GET_AMT_RET	Get the amount returned (not eaten) from user.
GET_REASON	Get the reason not eaten number from user.
GET_RATING	Get hedonic rating of the food item from user
GET_SALT	Get amount of discretionary salt added.
GET_DATA_C	Get the data collector code.
CHOOSE_FIELD	Have user describe a field for the template.
MAKE_TEMPLATE	Create a template from all possible fields.

4.3.4 Program Data Structures

Data is manipulated throughout the program using various data structures. The COD file exists external to the program in a text file. When the FOODDE program reads in this file, it does so into a doubly linked list. Each food item is stored in a node, and a node points to a previous and next node, so advancing forwards or backwards through the list is quite simple.

When food data is entered, the information is written to a food data node. An ordered insertion into the food data linked list is then performed on the node. The food data list is also doubly linked. Writing the food data to the food data file is accomplished by traversing from the start of the linked list, or the head, and visiting each next node until you reach the end of the list.

Both the Template file and the FOOD.DEF file are read into memory in the same way: Since they are defined as a RECORD of information in Pascal, you essentially read in one record. The different fields of information are filled according to their location in the record.

4.3.5 Program Operation

At startup, initialization is performed. The help arrays and other pertinent arrays are initialized, and the FOOD.DEF file is read in to obtain the default values. This file is an unformatted file that contains all the defaults in one record. If the FOOD.DEF file is missing or corrupted, it must be immediately recreated by the user. The main menu will now display. The user can choose from the six functions presently available. Depending upon what is chosen, that subroutine is then called by the main program. Usually, another menu will display offering the options associated with the selected function.

Whether the user chooses ENTER food data, or EDIT food data-ADD record, the same subroutine is called. A parameter is passed to let it know which it is doing, since it is of importance to know if there is food data in a linked list. New food data records are inserted using an ordered linked list insertion routine.

The other EDIT options, CHANGE and DELETE, require the user to enter a record number.

With CHAN(:, the record is located by searching through the linked list and then it is displayeo to the screen. The user now specifies the data field which they want to change by entering its associated position number, which appears on the screen. The appropriate subroutine for that fielo is the called for the user to enter the new value.

With DELETE, the record is located the same way as with CHANGE, and then displayed. The user is given the option now to delete the record. If deletion is confirmed, the node of that record in the linked list is unlinked and tossed away.

Choosing to Produce a Report will call the subroutine that will read in the food data file and then go through the link list, converting the active fields to their character equivalents and then placing them in the output line that will be written to the report.

If Create/Print Template is selected, a menu will display asking the user to specify which action is to be carried out. Creating a template involves prompting the user to see which fields he wants to activate for the template. At this time, the user may: choose the screen location for the field; make the field a default field; and specify high/low ranges for the field. This information will then be written to the template file. The template file is an unformatted file that contains all this information in one record.

Printing a template file involves reading in the template and processing the information stored in the record. The interpreted information is written to the output file.

Selecting Default Values from the Main Menu calls the subroutine that displays the defaults menu. Depending on what value the user wishes to change, the appropriate routine is invoked to obtain this information from the user. If the user makes a change to any of the defaults, the old FOOD.DEF is overwritten with new *values*.

4.4 Diet History Analyzer User's Guide

4.4.1 Introduction

The Diet History Analyzer is an interactive program on the VAX for creating and editing diet history files and then analyzing them to obtain nutritional information. This document describes how to use the Diet History Analyzer. Currently, it should be considered a "living" document. As the program is updated, this document will also be updated.

4.4.2 Getting Started

To run the Diet History Analyzer (DHA), several files are needed. All files, with the possible exception of the database files, must be in the s me subdirectory. Currently, the format and the content of the database files are being updated, so this will eventually change.

The source code files (ending with .FOR) and the object files (ending with

.OBJ) are not needed to run the program. The executable file (ending with .EXE) is the actual program. There are also several "resource" files (ending with .RSC). These files contains templates for the menus and are needed to run the program. Help files (ending with .HLP) are also used by the program for on-line help. Note that for every .RSC file there is a .HLP file. The following tables show the files needed to run the Diet History program:

Database Files	File Name Description
INGREDV6.DAT	Ingredients data file.
UNITSV6.DAT	Units of measure.

Program Files:	File Name Description
DIET.EXE	The main executable program.
DIETMENU1.RSC	The main menu template.
DIETMENU1.HLP	The main menu help.
DIETMENU2.RSC	Subject/Meal data menu template.
DIETMENU2.HLP	Subject/Meal data menu help.
DIETMENU3.RSC	Enter food item data menu template.
DIETMENU3.HLP	Enter food item data menu help.
DIETMENU4.RSC	Main editing menu template.
DIETMENU4.HLP	Main editing help.
DIETMENU5.RSC	Line editing menu.
DIETMENU5.HLP	Line editing help.
DIETMENU6.RSC	Line editing change line menu.
DIETMENU6.HLP	Line editing change line help.
EDSUBMEAL.RSC	Edit subject/meal data menu.
EDSUBMEAL.HLP	Edit subject/meal data help.
DANALYZE.RSC	The diet history analysis menu.
DANALYZE.HLP	Help for diet history analysis.

To start the program, type "RUN DIET" from the DCL prompt. When the program starts, a header with the version number will be displayed followed by a message indicating that the program is performing initialization. When the initialization is complete (approximately 2-3 minutes), the main menu will appear.

4.4.3 The Main Menu

The main menu currently contains 7 choices. To make selections, use the cursor keys or the RETURN key to move the selection bar up or down. Then, press ENTER on the numeric keypad to make the selection. Note that the selection bar wraps around from top to bottom and vice versa.

To get on-line help at any menu, press PF2. If you are using a VT-220 compatible terminal, or higher, you can also use the DO key instead of ENTER, and the HELP key instead of PF2. However, if you are using a terminal emulator on a DEC PC or other PC, the HELP and DO keys have different meanings.

The first item in the Main Menu is "CREATE a Diet History". Select this option to create a new diet history file. The next item is "APPEND to a Diet History". Select this option to add diet records to an existing coded diet history file. Regardless of which of these two items is selected, the actual coding process is identical.

The third item in the Main Menu is "CHECK a Diet History". This option will produce a listing file and optional printout of a coded diet history file in a readable format.

The fourth item in the Main Menu is "EDIT a Diet History". This selection will bring up a series of other menus for editing existing diet histories line by line.

The fifth item in the Main Menu is "COMBINE Diet Histories". This enables you to merge two different diet history files into one. The resulting file may be a new file, or may be one of the original files.

The sixth item in the Main Menu is "ANALYZE a Diet History". This will bring up a menu for diet history analysis. A report of the analysis can be printed without quitting from the program.

The last item in the Main Menu is "QUIT program". Select this option when you are completely finished. Since it takes 2 or 3 minutes to reinitialize, the program gives you a chance to change your mind before terminating.

4.4.3.1 Creating a Diet History File

After "CREATE" is selected from the Main Menu, you will be prompted to open the diet history file. Enter the file name, with extension, for the new file. For example, say the user typed "JUNK.DAT". If the file does not open properly, the user is prompted to retry, or cancel the operation and return to the Main Menu. If this is not the first file that has been opened in the current session, the previous file will appear as the default choice. You can select the previous file by just pressing RETURN rather than typing out the whole name.

After the file is opened, the Create Diet History menu will appear. This menu is used for entering the subject number, subject age, subject sex, date of consumption, time of consumption, and meal number (i.e., 1=Bkft, 2=Lunch, etc.).

Like other menus, use the cursor keys or RETURN to move the cursor from one field to another. Then, type into the field and press RETURN. Note: the contents of a menu field cannot be edited. The whole field must be retyped if you wish to make a change to it.

On-line help can be displayed by pressing PF2 key (or HELP key on some terminals).

When you are satisfied with all your entries, press ENTER (or DO on some terminals) to complete the menu. The next menu to appear is the Food Entry Menu.

Below the screen header there is a status box which shows the current Diet History file and the current line number within the history. There is also a line for messages which appear from time to time. This box will remain on the screen throughout the diet history coding process.

The first choice in the menu is "Enter a Food Item". When this is selected, the menu will temporarily disappear and you are prompted to enter the name of the food item. You can enter a partial name, or the full name if known. Upper and/or lower case may be used. After you type the food item name followed by RETURN, a list of all of the possible food items will appear. For example, if you enter "sugar", you will get a list of all of the different items that have "sugar" in their names such as "brown sugar", "maple sugar", "sugar substitute", etc, each with an associated number. You can then choose the number that corresponds to the exact food item you want. If there are more food items than can fit on the screen, the prompt at the bottom of the screen will indicate that you can press RETURN to see more choices. Some food items are identical up to sixty characters and you might need to see all eighty characters to make a decision on what food item to choose. This is done by entering the associated number of the food item, and then pressing PF3. The food item, in eighty character format, will display at the top of the food item list. Pressing PF1 will restart from the beginning of the list, and pressing PF4 will cancel the operation.

After you have chosen a food item, you will be prompted to choose the unit of measure being used such as pounds, grams, cups, etc. These choices will depend on the individual food item. Then, you are prompted to enter the amount of the food item in the chosen units. For example, you will be asked, "How many cups of brown sugar do you want?" You then enter the number and press RETURN.

Before entering the next food item, you may select "Change Subject/Meal data". This will cause the subject/meal entry menu to display, and you may now change any of the fields that are shown. For example, if you are entering data for all subjects and you come to data for a different subject, you will need to select this option to update the subject/meal data.

After all of the food item choices are made, the food Entry Menu will reappear. At this point, if you made a mistake, you could select "Delete last entry" to delete the last food item you entered: Each time this is selected, the last line in the diet history will be deleted and the status box will be updated accordingly.

When you are finished with the diet history, you can select "Complete current Diet History" to write out the information to the file, or "Cancel current Diet History" to cancel the whole diet history entry session. If you choose to cancel, you will be given a second chance to change your mind.

When entering the diet records into the diet history, order is not important. The diet history is sorted before it is written to the output file first by subject number, then by date of consumption, and then lastly by meal number.

When the diet history is completed or canceled, the main Menu reappears.

4.4.3.2 Appending to an Existing Diet History File

Appending to an existing Diet History file works functionally the same when you Create a Diet History (4.4.3.1). All new diet records entered will be appended to the ones that previously existed in the Diet History file, and then when the file is stored, all records be sorted.

4.4.3.3 Checking a Coded Diet History File

Once a coded Diet History file has been created, you may wish to check it. You can make a file and optional printout of a coded Diet History file in an easy to read format by selecting "CHECK a Diet History" from the Main Menu.

The "checked" diet history file will have the same name as the coded file except it will have the .CHK extension. Using the previous example for a data file, JUNK.DAT, the translated file would then automatically be named JUNK.CHK. If you just press RETURN on the second question, "no printout" will be assumed. When the translation is complete, the Main Menu will reappear.

As an example, lets have JUNK.DAT contain a one day diet history for one subject. Figure 4.24 is the checked file JUNK.CHK that was created using the check function.

4.4.3.4 Editing a Diet History

Once a coded Diet History file has been created, there are several ways to edit any of the diet records. It is helpful to have a printout of the checked Diet History handy (see section 4.4.3.2) during an editing session. After you select

"EDIT a diet history" from the Main Menu, you will be prompted to enter the name of the file to open. Then, the Diet History Editor Menu will appear.

The first choice in the menu is "Edit the Diet History". This is for editing the Diet History on a line by line basis. Section 4.4.3.3.1 contains an in-depth description of this portion of the program.

The second choice in the menu is "Duplicate this Diet History". When you select this, you are then prompted to enter the name of the file into which the entire diet history will be duplicated.

The final selection on the Diet History Editor Menu is "Exit Diet History editor". Selecting this option will return you to the Main Menu.

4.4.3.4.1 Line by Line Diet History Editing

Diet Histories can be edited on a line by line basis to insert and delete food item records, or to change a particular food item, amount, unit of measure, or any part of the subject/meal data. Selecting "Edit Diet History" from the Editor Menu brings up the Diet History Line Editor Menu:

The subject number, date, meal number, food item, and food item quantity fields of the first diet record of the diet history file will be displayed.

You can move through the diet history to find a specific line in three ways. First, you can advance through the line by selecting "Next line" from the Menu. Move the selection bar down to that command by using the down arrow key or the RETURN key. Then, press ENTER (or DO on some terminals) to advance one line. Notice that the line count in the status box is updated. You can continue to press ENTER until you reach the line you are looking for. The fields specified above will be displayed as they were for the first record.

Secondly, you can back up through the diet history by selecting "Previous line" from the Menu. This is done in the same way as with "Next line".

The third way to move through a diet history is to go directly to a specific line. If you have a translated listing handy, you may know exactly where you want to go, so select "Go to line _" from the menu. At the bottom of the menu, you will be prompted to type the line number to which you wish to advance.

You can also insert and delete lines in the diet history. To delete the current line that is displayed select "Delete this line" from the menu. To insert a line, you must select "Insert a line". You will be prompted to enter the subject/meal data, and then you must specify a food item, it's units, and a quantity. All lines are

FIGURE 4.24

```
**************************************************************************************************

                           CHECKING DIET HISTORY FILE:  JUNK.DAT

**************************************************************************************************

Line 01     1 21 M 890901 0900   1  01082      MILK, COW'S, LOWFAT; PAST & RAW, FLUID, 1% FAT
                             2.00 CUP                                                    488.0 Grams
Line 02     1 21 M 890901 0900   1  09206      ORANGE JUICE; RAW
                             2.00 CUP                                                    496.0 Grams
Line 03     1 21 M 890901 0900   1  43377      CEREAL, BRAN MUFFIN CRISP;
                             6.00 OUNCES                                                 170.1 Grams
Line 04     1 21 M 890901 0900   1  08010      CEREAL, CAP'N CRUNCH; (CORN W/OTHER GRNS)
                             2.50 CUP                                                     92.5 Grams
Line 05     1 21 M 890901 1200   2  86281      PIZZA WITH CHEESE TOPPING; BAKED FR HOME RECIPE, UNENR
                             3.0C SECTOR, 1/8 PIZZA                                      195.0 Grams
Line 06     1 21 M 890901 1200   2  14400      COLA SODA, COKE, PEPSI;
                             2.00 12 FL OZ CAN                                           740.0 Grams
Line 07     1 21 M 890901 1200   2  43330      POTATO CHIPS, PRINGLES; WHITE, RESTRUCTURED, LOW FAT AND SOD
                             1.00 CUP, STACKED (APPROX 20 CHIPS)                          43.0 Grams
Line 08     1 21 M 890901 1800   3  14096      WINE, TABLE, RED;
                             2.00 3.5 FL OZ GLASS                                        206.0 Grams
Line 09     1 21 M 890901 1800   3  13230      BEEF, SHORT LOIN, PORTERHOUSE STEAK; CHOICE, LEAN+FAT, BROIL
                             1.00 LB RAW AP                                              269.0 Grams
Line 10     1 21 M 890901 1800   3  11053      BEANS, SNAP, CREEN VAR; BOILED, DRAINED, WO/SALT
                             1.50 CUP, FROZEN OR CANNED                                  202.5 Grams
Line 11     1 21 M 890901 1800   3  85661      PIE, APPLE; BAKED, UNENR, VEG-S
                             1.50 SECTOR, 1/8 OF PIE                                     177.0 Grams

--------------------------------------------------------------------------------------------------
```

inserted at the end of the current diet history (the record will be placed in its proper position when it is sorted later).

If you want to change part of the diet record, select "Change this line". The Diet History Editor Change menu will display.

Once in the editor change line menu, if you select "Whole Food Item", you will be prompted to enter a food item, the unit of measure, and amount in the same way as in the program for creating a diet history (section 4.4.3.1). All the other choices allow you to change only one part of the food item data.

Select "Unit of measure" to only change the unit of measure for the item. Select "Amount of food item" to change only the amount consumed of the food item.

Select "Subject/Meal data" to change any of the subject/meal data fields. The new values will be entered in the same way as they were when creating a coded diet history (section 4.4.3.1).

When all changes are complete, select "Done with changes" to return to the Diet History Line Editor Menu.

When all editing is complete, select "Exit editor" from the Editor Menu. This will return you to the Main Menu.

4.4.3.5 Combining Diet History Files

Since it may be more practical at first to enter subject data at different times in different files, it may be necessary to merge several of these files into one larger file. This can be done by selecting "COMBINE Diet Histories" from the Main Menu.

You will be prompted to name the first diet history file. Then you are prompted to enter the name of the second diet history file. Finally, you are prompted to enter the name of the output file that will now be the combination of the two diet histories. The file specified may be a new name, or one of the input files which will then be overwritten.

4.4.3.6 Analyzing Diet Histories

Once a diet history file has been created, you may wish to analyze it for nutritional information. From the Main Menu, select "Analyze diet history" to bring up the diet history analysis menu.

In the first field, type the name of the coded diet history file to be analyzed. In the second field, type the name you want to give to the analyzed output report file. The normal convention for the output file is filename.REP.

If you enter a "Y" in the "Printout" field, the report file will be printed when the analysis is complete to the system line printer.

In the last field in the menu, you can list the nutrients you wish to show in the summary. At present, the maximum number of nutrients that can be contained in the report is 15. Any attempt to specify more than fifteen nutrients will result in an error (NOTE: The Diet History can be analyzed with a different set of nutrients to obtain information for more than fifteen nutrients). To see what numbers correspond to what nutrients, type a question mark in the field (?) and press ENTER. The help file will be displayed in two screens.

For example, suppose you want your report to contain information on only water, calories, protein, and fat. These are listed as nutrients 1 through 4. So, you could type the following into the menu field: 1-4

Suppose in addition you want to show sugar. Sugar is listed as nutrient number 7, so you could type the following: 1-4,7

Appendix A is contains a chart correlating the nutrient numbers and abbreviations with the actual nutrient names. The units of measure for the nutrients are also included.

Once you have filled in all of the menu fields as you want, you may press ENTER to start the analysis, or PF4 to cancel and then return back to the Main Menu. Whatever you type into the Diet History Analysis Menu field will be retained for the next time in an external defaults file. This prevents having to retype every field the next time you run the analysis.

After you press ENTER to start the analysis, a status box will appear at the bottom of the menu. This will show you which diet history is currently being analyzed. The word "Analyzing" will be blinking to indicate the process is executing.

When the analysis is complete, a message will indicate that it is complete and that the report is being printed (if selected). Then, the program will automatically return to the Main Menu.

On the following page is the analyzed report file for the file JUNK.DAT that was checked in section 4.4.3.2.

Notice that the report lists a "?" for some nutrient values of a food item. This means that there was no information for this nutrient in the food item, and so it is considered missing data. Since missing nutrient data may result in underestimating the totals, an asterisk (*) is used in the report to flag a total that contains missing values.

```
S U B J E C T:    1                    T O D A Y S  D A T E: 12-SEP-89                                    P A G E  :001
D A T E       : 890901
A G E         : 21
S E X         : M

 FOOD ITEM             H2O    CAL   PROT    FAT    CHO     C      THI    RIB    NIA    B6     FOL    B12     P      MG     FE
-------------------------------------------------------------------------------------------------------------------------------
MILK, COW'S, LOWFAT;  439.6  204.3  16.1    5.2   23.4    4.8    0.2    0.9    0.5    0.3   24.9    1.8  469.5   67.5    0.3
ORANGE JUICE; RAW     438.0  223.3   3.5    1.0   51.6  248.1    0.5    0.2    2.0    0.2  150.3    0.1   84.4   54.6    1.0
CEREAL, BRA[illegible]    4.3  558.0  14.2    4.3  133.4    2.6    2.3    2.6   30.1    3.0   17.1    9.0  811.4  326.6   27.1
CEREAL, CAP'N CRUNCH    2.4  389.5   4.8    8.6   74.9    0.1    1.7    1.8   21.6    2.6  594.8    5.9  116.6   38.0   24.6

MEAL   1 TOTAL        884.3 1375.1  38.6   19.1  283.3  255.6    4.7    5.5   54.2    6.1  787.1   16.8 1481.9  486.7   53.0
*******************************************************************************************************************************

PIZZA WITH CHEESE TO   94.2  460.3  23.5   16.2   55.2   15.7    0.2    0.4    2.0     ?      ?      ?   380.3     ?     2.0
COLA SODA, COKE, PEP  661.6  303.5   0.1    0.1   77.0    0.1    0.1    0.1    0.1    0.1    0.1    0.1   88.9    7.5    0.3
POTATO CHIPS, PRINGL    0.9  223.7   2.9   12.5   25.0    5.6    0.1    0.1    1.5    0.3   19.4    0.1   73.2   24.6    0.7
                                                                                      *      *      *             *
MEAL   2 TOTAL        756.7  987.5  26.5   28.8  157.2   21.4    0.4    0.6    3.6    0.4   19.5    0.2  542.4   32.1    3.0
*******************************************************************************************************************************


WINE, TABLE, RED;     182.4  148.4   0.5    0.1    3.6    0.1    0.1    0.1    0.2    0.1    4.2    0.1   28.9   26.8    0.9
BEEF, SHORT LOIN, PO  140.9  804.4  67.6   57.1    0.1    0.1    0.3    0.6   11.1    1.0   18.9    5.9  508.5   67.3    7.2
BEANS, SNAP, GREEN V  180.7   70.9   3.9    0.6   16.0   19.7    0.2    0.2    1.3    0.2   67.5    0.1   79.0   50.7    2.6
PIE, APPLE; BAKED, U   84.3  453.2   3.9   19.7   67.5    1.8    0.1    0.1    0.8     ?      ?      ?    39.0     ?     0.6
                                                                                      *      *      *             *
MEAL   3 TOTAL        588.3 1476.9  75.9   77.5   87.2   21.7    0.7    1.0   13.4    1.3   90.6    6.1  655.4  144.8   11.3
*******************************************************************************************************************************

                                                                                      *      *      *             *
DAY TOTAL            2229.3 3839.5 141.0  125.4  527.7  298.7    5.8    7.1   71.2    7.8  897.2   23.1 2679.7  663.6   67.3

                                                                                      *      *      *             *
AVG   1 DAY INTAKE   2229.3 3839.5 141.0  125.4  527.7  298.7    5.8    7.1   71.2    7.8  897.2   23.1 2679.7  663.6   67.3

  ?  indicates missing nutrient data
  *  indicates values may be underestimated due to missing nutrient information
```

4.5 Diet History Analyzer Programmer's Guide

4.5.1 Introduction

The Diet History Analyzer (DHA) is an interactive program on the VAX for creating and editing coded diet history files and then analyzing them for nutritional information. This document describes some of the details on how the program works.

4.5.2 Program Files

The Diet History Analyzer consists of several Fortran source code files, the Menu Manager source code files, the link file, the resource files (for menus), the help files, and a couple other data files. The following is a list of the source code files used with a brief description of each:

DIET HISTORY SOURCE CODE

DIET.FOR	- Startup, Main menu, coding functions, etc.
DODIET.FOR	- Diet processing;ingredients, units, check,etc.
EDITDIET.FOR	- Edit routines to edit coded diet history.
DANALYZE.FOR	- Analysis start up and menu.
DANA.FOR	- Main analysis routines.
REPORT.FOR	- Report generating routines.
INGRED.FOR	- Obtain food item information.

MENUS & GRAPHICS SOURCE CODE

MENUS.FOR	- Menu handling functions.
GRAPHICS.FOR	- Screen line graphics (boxes, etc.)
DIETUTILS.FOR	- Diet history specific utilities.
UTILS.FOR	- Utilities (get tokens, etc.)
DATEUTILS.FOR	- Date & time functions.

RESOURCE FILES

DIETMENU1.RSC	- The main menu template.
DIETMENU2.RSC	- Subject/Meal data menu template.
DIETMENU3.RSC	- Enter food item data menu template.
DIETMENU4.RSC	- Main editing menu template.
DIETMENU5.RSC	- Line editing menu.
DIETMENU6.RSC	- Line editing change line menu.

EDSUBMEAL.RSC	- Edit subject/meal data menu.
DANALYZE.RSC	- The diet history analysis menu.

<u>HELP FILES</u>

DIETMENU1.HLP	- The main menu help.
DIETMENU2.HLP	- Subject/Meal data menu help.
DIETMENU3.HLP	- Enter food item data menu help.
DIETMENU4.HLP	- Main editing help.
DIETMENU5.HLP	- Line editing help.
DIETMENU6.HLP	- Line editing change line help.
EDSUBMEAL.HLP	- Edit subject/meal data help.
DANALYZE.HLP	- Help for diet history analysis.

<u>MISCELLANEOUS</u>

DIET.FIL	- Contains names of database files.
DIET.LNK	- Project link file.

4.5.3 Subroutines & Functions

The following section shows each of the subroutines and functions in each of the source files for the Diet History Analyzer (NOTE: DIETUTILS are discussed here since they are not discussed in the RECIPE CODER/EDITOR/ANALYZER documentation):

<u>DIET.FOR</u>	- Startup and main menu features.

<u>DODIET.FOR</u>

DODIET	- Recipe coding menu.
R_JUSTIFY	- Right justify a character string.
L_JUSTIFY	- Left Justify a character string.
SYS_SORT	- VAX system sort of the diet history file.
PROCESSDIET	- Edit diet history menu.
DOINGREDIENT	- Choose a food item from ingredient file.
DELETERECORD	- Delete the last diet record entered.
DIETDONE	- Write scratch file out to diet history file.
OPENDIETFILE	- Open the diet history file.
ISCAN	- Find matching Ingredients in database.
FILE_OUT	- Write out coded diet history scratch file.
MISSING	- (future use) Log missing ingredients.

STARTUP	- Read in ingredients & other init.
READREAL	- Input a real number.
CHECK	- Translate coded diet history file.

EDITDIET.FOR

DIETEDIT	- Get file to edit, display menu.
DUPLICATEHISTORY	- Duplicate a diet history to new file.
READINDIET	- Read diet history file into common arrays.
WRITEOUTRECIPE	- Write contents of common array to file.
EDITLINES	- Menu for line editing.
NEXTPREVLINE	- Advance/Backup in diet history one line.
CLEARMESSAGE	- Clear message line in the status box
GOLINE	- Go directly to a diet history line.
GETSUBMEAL	- Get subject and meal data.
DELETELINE	- Delete a line from the diet history.
INSERTLINE	- Insert a line/diet record into diet history.
CHANGELINE	- Menu to select what part of line to change.
CHANGESUBJECTMEAL	- Menu to change subject/meal data.
CHANGEWHOLEINGRED	- Edit whole food item line.
CHANGEUNIT	- Edit the unit of measure.
CHANGEAMOUNT	- Edit the amount of the food item.
COMBINEDIETFILE	- Get files to combine.
COMBINEDIET	- Combine diet histories.

DANALYZE.FOR

MYSET_UP	- Initialize and call the analysis menu.
ANALMENU	- Handle the analysis menu.
READ_MYDEFS	- Read old parameters for the menu.
CONVERT_NUTNUMS	- Parse nutrient number string.
HELP_NUTS	- Display nutrient label help.
WRITE_MYDEFS	- Write out new menu defaults to file.
OPEN_REPFILES	- Open all files needed for analysis.
SET_INIT	- Calculate report parameters.
ASSIGN_NUTLABELS	- Assign nutrient labels to an array.
COPYSTRING	- Copy string, pad w/blanks, NULL terminate.
MENUSTAT	- Put menu status message at menu bottom.

DANA.FOR

DIET_ANA	- Start analysis and read diet history lines.
READ_RECLINE	- Read a line of the diet history.
INGREDIENT	- Process ingredient food item line.
CVT_NUM	- Converts a string to real number.
ADD_UP	- Add contents of one array to second array.
MEAL_TOTALS	- Report meal totals, add to day totals.
DAY_TOTALS	- Report day totals, add to running average.
AVG_INTAKE	- Calculate average daily intake, report it.
ROUNDUP	- Round a real number to the tenth's digit.

<u>REPORT.FOR</u>	
REPORTIT	- Generate report file.
MAKE_FORMAT	- Create format string for report.
OUT_STRING	- Character string of nutrient values.
<u>INGRED.FOR</u>	
GETINGREDIENT	- Ask user for food item name.
GETUNIT	- Ask user for unit of measure.
GETMULT	- Ask user for quantity of food item.
USCAN	- Handle unit choices.
<u>DIETUTILS.FOR</u>	
POSTMESSAGE	-Put message to message line in status box.
INITSTATBOX	- Put up status box.
INITMAINRES	- Get main menu and other resources.
CHOICE	- Handle a numbered list of choices.
EOS	- Find end of non-NULL-term string.
UPDATELEVEL	- Update level number in status box.
UPDATELINE	- Update line number in status box.
ASK	- Get an answer to a prompt question.

4.5.4 Program Operation

After starting the program, initialization is performed. This initialization can take 3 or 4 minutes because it reads the entire ingredients database into memory. The ingredients database is a Fortran keyed indexed file which must be prepared in advance with the Dumptable program.

After the initialization, the Main Menu appears. The user can then choose which function to perform. The structure of the code that handles the Main Menu (and any other menu) is covered in more detail in the Menu Manager document. Depending on what the user chooses, the main routine calls the appropriate subroutine. That subroutine may in turn put up another menu. In that way, menus can easily be nested. When the function is completed, control is returned to the main routine and the Main Menu is redisplayed.

The "Create a Diet History" and "Append to a Diet History" options from the Main Menu use the same subroutine. A parameter is passed to that subroutine to indicate whether the coded diet history file should be opened as 'NEW' or 'OLD'. Other than that, the rest of the coding is the same for both choices.

While the user is coding, the diet history is being composed in a temporary scratch file. This allows for the possibility of deleting diet record lines before committing them to the diet history file. When the diet history is complete, the lines are copied to the diet history file. If the user cancels, the scratch file is closed and discarded without updating the diet history file.

If the user chooses "Check a Diet History" from the Main Menu, a subroutine is called which reads the coded diet history file line by line and translates it into a more readable format. The translated file is a regular ASCII text file and it can be printed directly from the program to the default printer.

Diet History editing is somewhat more complicated. When this option is chosen from the Main Menu, another menu appears. The user can choose to duplicate a diet record or edit a diet history on a line by line basis. If duplication is selected, the user is prompted for the new diet history file name. The entire diet history is read into an array (one line per array element), and then the whole diet history is written out to the new diet history file.

When the user chooses to edit the diet history on a line by line basis, the entire diet history is read into an array in the same way as with duplication, but it is also deleted from the coded diet history file. This is necessary in order to enable the file to be updated later. The actual editing is performed on the array, not the file. When the editing session is complete, the array is written out to the diet history file.

4.5.5 Diet History Analysis

The diet history analysis is fairly simple from the user point of view, but it requires some explanation from the programming point of view. When the user chooses "Analyze Diet History" from the Main Menu, the Analysis Menu appears. The user types the name of the file to be analyzed, the report file name, and the nutrients that they wish to appear in the report. When the user types ENTER, the analysis will proceed automatically.

Each line from the coded diet history file is read in one at a time. The food item code number is used as an index into the database ingredients file to locate its nutrient values. The nutrient values are stored for 100 gram weights of the food item, so they are adjusted according to the amount of the food item that is read in from the coded diet history file. One array is used to keep track of the nutrients of food items from one subject, with the same date and meal number. This will be the total meal nutrients. The file is read line by line, the ingredients are found in the database and the meal total is calculated.

From these meal totals, the day total will be calculated. The day total is the sum of all of the meal totals for a subject on one date. From the day totals, an average daily intake of nutrients can be calculated for a subject. The average daily intake is the sum of the day totals divided by the number of different days for which there is subject data.

4.5.6 File Formats

The DHA uses several input files: the ingredients database and units of measure database. It also creates and reads coded diet history files. All of these files are Fortran keyed indexed files. In a keyed indexed file, all records are of fixed length and a certain field is designated as the "key". The key field is just a certain range of character positions. For example, if a keyed indexed file contains records of length 80, we might specify that characters 1 through 10 are to be the key. Each record must have a unique key, and Fortran automatically maintains the file in the order sorted on the key.

To open an existing keyed indexed file, the following Fortran OPEN statement would be used. Note that the words in capitals are typed exactly as shown, whereas the words in lowercase italics are variable:

```
OPEN (UNIT=lu, FILE="filename", STATUS="OLD",
          ACCESS="KEYED",ORGANIZATION="INDEXED",
          KEY=(start:end:datatype),RECL=recordlength,
          ERR=linenum)
```

The unit number, specified by lu, can be whatever logical unit the program is using for the file. The start and end numbers specify the starting and ending character positions in each record to be used as the key field, and the datatype is the type of information in the field such as INTEGER, CHARACTER, REAL, etc. In the DHA, all keyed indexed files have CHARACTER keys. The recordlength is the number of bytes (or characters) in each record, and the linenum is the program label number to transfer to on error conditions. At that point, the program can display the appropriate error messages and take whatever action is necessary. To create a new keyed indexed file, the status would be changed from "OLD" to "NEW".

The ingredients file contains the entire food item nutrient database. The file is prepared in advance by running the Dumptable program. Dumptable translates the Oracle table into the Fortran format. Once it is read into memory, accessing it is very fast. Note that Dumptable only needs to be run when a major update is performed on the Oracle table. The ingredients file is opened to logical unit 15 with the following OPEN statement:

```
OPEN (UNIT=15, FILE="INGREDV6.DAT", STATUS="OLD",
          ACCESS="KEYED", ORGANIZATION="INDEXED",
          KEY=(1:10:CHARACTER), RECL=165, ERR=990)
```

In the record format for the ingredients file, there are two important things to notice:

The ingredient name field length is 80 characters and that the sum of the number of bytes in one record exceeds the number declared in the file OPEN statement, i.e. 165. This is okay as long as every OPEN statement, including the one in the Dumptable program, is consistent. Fortran will just allocate whatever extra amount is needed. For historical reasons, 165 is retained to avoid incompatibilities.

The unit of measure file is also prepared in advance by running Dumptable. It is opened to logical unit 1 in exactly the same way as the ingredients file except that the record length is 420. This record length is not long enough, but for backward compatibility, it is retained.

In addition to the database files, the DHA also creates keyed indexed files for the coded diet histories. These files are used internally by the program. The user would normally translate the file into a readable format before examining the coded diet histories. In this case, the number of bytes in each record is exactly the same as the number declared in the OPEN statement, i.e. 115. The key for the diet history file is the line number, and it is located in the first field of the file. See Appendix E for more information concerning the coded diet history file.

There are also two other types of files: Resource files and the Help file. Resource files contain the templates for menus. There is one resource file for each menu. For more detail on the format of resource files, see the Recipe Menu Manager document. Help files are just regular ASCII text files. There is one help file for each resource file. Help can be requested by the user at any menu. When the user presses the PF2 key (or HELP key on some terminals), the help file is read in and displayed on the screen.

4.6 Universal Data Entry Program

The Universal Data Entry program (UDE) allows the user to build a simple data file, add to this data file at a later date, edit records in the file, and produce a report from the file. To run the data entry program, simply go into the directory in which you wish to place the data file and type UDE followed by <RETURN>. The Main Menu will appear and you can then select your option. Detailed explanations of the files created by this program are included in Appendix F.

THE MAIN MENU

The Main Menu has a total of eight options:

1. Enter Data Into a New File - With this option you create a new data file. The file name you specify for this data file cannot previously exist.

2. Enter Data into an Existing File - This option is used to continue

entering data into a file that already exists. NOTE - the data file must have been created by this program.

3. Edit An Existing Data File - Allows you to edit specific records of an existing data file that was created by this program.

4. Produce a Report From an Existing File - Allows you to produce a formatted printout of the data file.

5. Add New Variables to an Existing File - This option allows the user to add new variables to an existing data file. The values of the new variables are appended to the end of the appropriate records that already exist in the data file.

6. Rearrange the Variables in the Data File - Allows changes in the order in which the data is listed in the data file.

7. Change the Size of Numeric Variables - Allows you to enlarge numeric variables in an existing data file. The data file is automatically updated to conform to the new definition of the data.

8. Quit - Exits from the program.

4.6.1 Enter Data Into A New File

When you first sit down with new data, the first thing you should do is choose option 1 on the Main Menu. This option allows you to create a new file to store the data you will enter.

After you choose option 1, a sub-menu will ask you to verify that you really wish to enter data into a new file. If not, choose sub-option 2, which will bring you back to the Main Menu.

After choosing sub-option 1, you will be asked to enter the name of the new file. The file name can be any legal VAX file name but should be limited to a maximum of 17 characters. Be sure NOT to include the dot extension (i.e., .DAT) after the file name. This will be supplied by the program. If you do include it, nothing adverse will occur, but you will have to enter it again.

If by accident you enter the name of a file that already exists within that directory, the program will inform you of your error and return you to the Main Menu so that you can choose another option.

After you enter a correct file name, you will move on to the variable definition section of the program. It is here that you will create the variables that will identify the data that you are entering. The variables are split into two families: grouping variables and test variables.

Test variables are the variables that are being measured, recorded, and studied. They might include such things as height, weight, age, sodium intake, blood cholesterol level, etc. To be able to distinguish among different values, you need some means of grouping the values. A sodium intake value does not mean much if we do not know when, where, or from whom it was taken. This is where the grouping variables come in.

Grouping variables can include subject number, date, group, location, or any other information that is needed to uniquely identify each record in the data file.

In the following example, ENERGY, SODIUM, and FAT are the test variables. They were measured and will be examined in future analyses. SUB, GROUP, and DATE are the grouping variables, for they identify each record of data. A value of 3000 in energy may not be unique, but it can be identified as belonging to subject 1 of the MRE group on day 12, which is unique.

EXAMPLE

SUB	GROUP	DATE	ENERGY	SODIUM	FAT
1	MRE	12	3000	3400	485
1	MRE	13	2830	3900	621
1	RCW	12	2575	4200	380
1	RCW	13	2800	3700	420
2	MRE	12	3100	3500	560
2	MRE	13	3234	3600	740
3	RCW	12	2800	4000	630
.	.	.	.	.	.
.	.	.	.	.	. . . .

The first family of variables that you will define are the grouping variables. First you will be asked to enter a variable label. This label can be anything you wish, but should relate to the data it will represent, i.e., SUBNUM for subject number or ENERGY for energy intake. The variable names should be limited to eight characters.

Next, a small menu will appear inquiring as to the type of variable (integer, real, or character). If the values for that variable will only be numbers without decimals, then choose option 1, integer. If they will be numbers with decimals, then choose option 2, real. If they will only be strings of alpha-numeric characters, then choose option 3, character.

Lastly, you will be asked the length of the variables. For characters, the length will be the number of characters in the longest string. If the longest string value is "carbohydrate" for that variable, then the length for that variable will be 12. For integers, the length will be the number of digits in the largest number. If the largest number is 1024, then the length of the variable is 4. For real numbers, the length will be the number of digits before the decimal plus the number of digits after the decimal plus 1 for the decimal, so that 123.01 has a length of 6. For real numbers, you will also be asked for the number of digits after the decimal. In the last example, it would be 2.

After the length or number of digits is calculated, it is best to increment it by one to cover any unexpected number that might pop up. If you are sure you calculated the correct longest length that will ever be entered for that variable, then you will not need to add the extra space.

After you enter the length, you will be asked for the label of the next grouping variable. When you have entered the last grouping variable, just press <RETURN> when the program asks for the next one, and you will move on to the definitions of the test variables. The process for the test variables is exactly the same as the one for the grouping variables. When you are finished and the program asks for another one, press <RETURN> to continue on.

Now that you have defined the variables, you have a chance to make corrections before you go on to the actual data entry. There are five options you can perform:

(1) Delete a Variable - If for some reason you no longer want a variable, choose this option. You will be asked for the variable number that is listed next to the variable that you wish to delete. Enter the number or 0 if you wish not to delete any, then press <RETURN>.

(2) Exchange Two Variables - If you want to reorder the variables, you can specify them with this option. When you enter data, the program will ask you for the data in the order that the variables are listed on the screen, so they should be listed in the same order as you want to enter them. To reorder two variables, choose option 2 and enter the numbers of the variables when asked. The program will then switch the two variables in the list. If you change your mind, just enter 0 for either ariable number and no changes will be made.

(3) Edit a Variable - Option three allows you to edit one of the variables by entering the number of the variable. If you change your mind, enter 0 and no variable will be edited. After choosing one of the variables, you can change its label, type, or size by choosing the appropriate sub-command, then following the directions given. Sub-command four allows you to leave the variables as is.

(4) Continue Entering Variables - If you forgot to define that last variable, just choose option four to resume entering variables. It will follow the same format as before except that now you will also be asked whether or not each variable will be a grouping variable. Enter a Y if it is, an N if it is not. When you are done defining variables, press <RETURN> and you will return to this option of the menu.

(5) Finished, Go On to Data Entry - This is it! You've finished with all variable definitions and you think that everything is correct. To delve into the fantastic world of data entry, just choose option five and away you go!

Once you begin data entry, it is all pretty straightforward. The program will ask you, in order, to enter data for each of the variables. One cycle through the complete variable list constitutes one record, which will then be saved to the data file. Enter the value for each of the specified variables and press <RETURN> after each one. After you've gone through all of the variables of one record, you are given the opportunity to check your input. If you wish to check for mistakes, enter a 1. If you want to continue, just press <RETURN>.

If you enter a 1, a menu will appear with three options:

(1) Continue On - The values for that record are all correct. Continue with the data entry.

(2) Reenter a Value - If one of the values is wrong, choose this option to correct it. You will be asked which variable is incorrect, and then you will be asked for the correct value.

(3) Delete Entire Record - If there are too many mistakes, or the entire record is wrong, this option allows you to delete it all. You will then be asked whether or not you have more data to enter. If you do, enter a Y, if not, enter an N.

If you do not enter a 1 and just press <RETURN>, you will immediately proceed to the next record to be entered. When you are done entering the data, or you just want to stop for now, enter a -9 or a <RETURN> to exit, depending upon which one is displayed on the current variable. You can exit at any point during the data entry, but if you exit half-way through a record, the part that you entered will not be saved. Only complete records are saved to a file so it is best to break after you complete a whole record.

Once you choose to exit, you are given one chance to change your mind. If you have more data and want to continue, enter a Y. If you really want to exit, enter an N. If you choose to exit, you will return to the Main Menu where you can choose another option.

When you exit to the Main Menu after creating a new data file, three files will be saved in the current directory. Filename.DAT (where Filename equals the name you have chosen for the file) will contain the actual data that you have just entered. Filename.FMT will contain the variable formats that you defined for the data so that you will not have to reenter them again when you want to add more data to the file. NOTE - Filename.FMT is a file of pascal records and is unreadable to you, so don't bother trying to edit or print it. The third file is Filename.TXT, which contains a list of the variables you defined, their FORTRAN formats, and a list of the columns that they are in. This file will prove useful for setting up the job control language (JCL) to statistically analyze the data.

Of the three data files, Filename.DAT and Filename.FMT MUST NOT BE DELETED. If you delete Filename.DAT, you will obviously lose all of the data that you have entered. If Filename.FMT is deleted, you will lose the formats for your variables, and you will be unable to enter more records into the file, edit the file, or produce a report from the file. Filename.TXT is not essential to the running of the program and can be deleted if not needed.

4.6.2 Enter Data Into An Existing File

If for some reason you did not finish entering your data into the data file in one sitting, this is the option you would choose to continue entering data. The file name you specify to continue entering data into MUST be a file that was created by this program. If it is not a file created by this program, then there will be no format file for the variables and the program will not run.

After you choose option 2 on the Main Menu, a sub-menu will ask if you really want to enter data into an old file. If not, enter a 2 to return to the Main Menu. Otherwise, enter a 1 to continue on to data entry.

If you entered a 1 to continue, you will be asked for the file name. Again, enter the file name without the dot extension. If you enter a correct file name, you will go straight to the data entry section of the program. You do not need to re-define the variables because their formats were saved when you first created them and are automatically loaded by the program.

The rest of the data entry runs exactly the same as entering data into a new file. There are no differences. If you do not know how this part of the program works, read the section on Entering Data Into a New File.

4.6.3 Edit An Existing Data File

After the data file has been validated and errors have been found, you can choose option 3 on the Main Menu to edit the data file. After you choose this option, you will be asked whether you wish to edit. If not, enter a 2 to return to the Main Menu. Entering a 1 will allow you to edit a file.

After entering a 1, enter the valid file name of a file that exists and was created by this program. Please do not include the dot extension or else you will be asked to do it again.

Once a valid file name has been entered, you will be asked which record to edit. If you change your mind about editing or are finished editing enter a 0 at the record number prompt to return to the Main Menu.

This is where you need a report of the file, for the report contains the record numbers you will need to edit the file. If you have not yet produced a report, return to the Main Menu and choose option 4. If you do not know how to produce a report, read the section on Producing A Report From A File.

On the printout of the report, the record numbers are listed in the column headed by REC. They begin with the number 1 and increment by 1 until the end of the file is reached. When the program asks for a record number, enter the number of the line in which a correction needs to be made. If the third line has an error, then 3 is the record number that should be entered for that line.

Once you have chosen a record number, the contents of that record are displayed. Each variable is listed along with the values in each variable.

You have four options:

(1) Continue On - If all the values have been corrected, or if you have chosen the wrong record, then choose this option to continue on. All corrections will be saved in the file. After choosing this option, you will be asked for another record number.

(2) Reenter a Variable - If one of the values is wrong, choose this option to correct it. Enter the number that is to the left of the variable that you wish to correct. You will then be asked for the new value. Enter the value and press <RETURN>. The variables will be listed again, along with all updated values. When all changes have been made, press <RETURN> and you will again be prompted for a record number.

(3) Delete a Record - If the entire record is a mistake, choose option 3 to delete it from the file. NOTE - use this option only after all other editing has been completed because the remaining records are renumbered immediately. The record numbers on the printout may no longer correspond to the record numbers in the file. When you start deleting records, start with the highest record number and work your way to thc lowest to be sure the record numbers do not change for following records. Always check the content of the record before you

delete it to make sure that it is the record you wish to delete.

(4) Renumber a Record - This option will allow you to reposition the record within the data file. You will be asked if you really want to renumber the record. If you do, enter a 1. Entering a 2 will exit without making any changes.

If you choose to reorder, you will be asked for the new record number. If you want to make the record the fourth one in the file, enter a 4. The record will become the fourth record in the file, and the other records will be pushed up by one. NOTE - use this option only after all other editing has been completed because renumbering one record causes the remaining records to be renumbered immediately. The record numbers on the printout may no longer correspond to the record numbers in the file.

No new files are created by editing. The old data file is replaced by the new one containing the corrected values. A new report should be generated from the updated file.

4.6.4 Produce a Report From An Existing File

If you need a hard copy of the data file for validation or any other reason, then choose option 4. This option allows you to produce a report from the data file and gives you some control over the format of the printout.

After choosing option 4, a sub-menu will ask you to confirm your choice of producing a report. If you chose option 4 by accident, then choose sub-option 2 to return to the Main Menu. If you want to produce the report, then choose sub-option 1.

You will now be asked for the file name. The file name must be the name of a file that exists and was created by this program. Do not forget to exclude the dot extension.

Once you have entered a valid file name, a list of the variables will be displayed along with some other information. An example of such a listing is as follows:

```
#) REC      Rec      3     1) SUBJECT SU          2
2) DATE     DATE     9     3) ENERGY  ENERGY      7
4) PROTEIN  PROTEIN  7     5) FAT     FAT         7

   Report Width = 41 Characters.
```

The first thing you will probably notice is that there is a variable listed that you did not define, REC. This variable stands for the record numbers in the file and is used by the program to identify specific records in the file. The first record in the file has a record number of 1, the second has a record number of two, and so on. The record numbers are included in the report because they are needed if you wish to edit any records at a later date.

Next to the variable label, you will see a copy or a partial copy of the label. This copy is the heading that will appear over the column of the report containing the values for that variable. The default headings are the labels of the variables. Some of them may be truncated, such as SUBJECT in the example above, because they are made to fit within the width of the variables you specified during variable creation. Since SUBJECT was defined as being two characters wide it will take up only two characters in the printout. Only the first two characters of SUBJECT, SU, will fit in that column.

The numbers listed are the column widths of those variables. A total is given showing how many characters one line of the printout will contain.

A sub-menu of three options is present. If you've gotten this far and decide that you don't want to produce a report after all, choose option three to return to the Main Menu, and no report will be produced. If you want to produce a report with the default headings and widths listed, choose option one, and the report will be generated.

If you wish to make changes to the headings, choose option two to change the default headings. When you choose this option, you will be asked which one you want to change. Simply enter the number next to the variable label to choose one, or a 0 if you change your mind and don't want to change any. As you can see, REC has no number next to it, only a # symbol, so you cannot alter this heading.

If you want to change the heading of SUBJECT, you would enter the number 1 for that variable. You will be asked if you want to change the column heading, the column width, or change your mind. If you change your mind and decide to leave the heading as is, enter a 3. If you want to change the width so that the whole label will fit into the column, choose option 2. The program will ask you for the new width, and in this example you should enter a 7. The variable list will be shown again, and this time the SU is replaced by SUBJECT, and the length of the entire report line is increased by 5 to 46 characters. You can change the width to any reasonable size, but there is one limitation: you cannot change the width to a size smaller than the width you allotted for the data when you created that variable. You, therefore, could not change the width of SUBJECT to 1 because that would not accommodate the width of the data in the SUBJECT variable, which was designated as two characters in the example.

You can change any of the headings by choosing option 1 after selecting which variable heading to change. You will then be asked what the new variable heaoing will be. You can use this option to change the heading of variable DATE to DAY. If you change a heading, the variable label itself will not change, as you can see when the list is re-displayed on the screen. Only the heading on the printout changes. The data is not affected by formatting the report.

Once you have formatted the heading to your liking, you can then choose option 1 to produce a report with the new headings. Choosing option 1, will return you to the Main Menu. The more records in the file, the longer it will take to produce the report.

After the report is produced, two new files will exist in the directory: Filename. REP (where Filename = the name you entered), which contains the actual report, and Filename. DFT, which contains the new default headings for the report, so that the next time you choose to produce a report, the headings will be loaded in for you.

Filename.DFT is a pascal record file, so it can't be printed or edited. It is not a necessary file, so it can be deleted, but you will lose your customized headings and the next time you produce a report, the variable labels will be the default headings again.

To print the report, choose option 8 on the Main Menu to quit the program and type PRINT Filename.REP at the DCL prompt, where Filename is the name of your data file. Your report will then be printed on the system line printer. Once the file is printed, you can delete the report file.

The printout will also contain the file name used to generate the report and the date and time the report was created. Even though the number or characters per line on a page can only be 132, you can produce a report that contains up to 255 characters per line. If your report exceeds 132 characters per line, it will be split into two reports, and both will be sent to the same file.

4.6.5 Add New Variables to a File

After you finish entering all of the data, someone may decide that they want to include a new variable, say cholesterol, that was not included in the old data file. You could create a new data file, define the variable for cholesterol, enter the values, and try to merge the new file with the one containing the rest of the data. This would work, but you could not use the UDE program to do any further editing or report producing on the new combined data file because the variable formats would be in separate files.

To be able to add new variables to an existing file and still be able to edit and produce reports, choose option 5 on the Main Menu. This option allows you to define the new variables, enter the new data, and automatically merge the new data.

A sub-menu will ask if you want to proceed with entering new variables. A response of 2 to this question will bring you back to the Main Menu; a 1 will allow you to continue on.

After choosing 1, you will be asked to provide the file name of the existing data file that was created by this program. Do not include the dot extension.

If you provide the name of a valid file, a new sub-menu will be displayed. This sub-menu provides you with three options:

(l) <u>Enter New Variables to a File</u> - If you have not begun to enter the new values and have not defined the variables for those values, then choose this option. If the file you selected already has new variables that are in the process of being entered, you will be warned and will be given the choice of returning to the Main Menu or to continue on with creating new variables. If you do continue on, some information may be lost concerning the current state of each record in the file. More on this later.

(2) <u>Continue Entering Values into New Variables</u> - If you have defined the new variables previously but exited the program without completely finishing the data entry, you can continue entering data with this option. If the file you chose does not have new variables that have been already defined, you will be returned to the Main Menu.

(3) <u>Return to the Main Menu</u> - If you change your mind about creating or entering new variables, choose this option and you will be returned to the Main Menu without any changes being made.

The first time you sit down to enter new variables, you will need to choose option 1. You will receive the message "Creating tree...." The program is now loading information about the records in your data file into memory. Once it has been loaded, you will be asked to define the new variables in exactly the same way as when you chose to enter data into a new file. Notice that now you are asked only for test variables, not grouping variables. You are not able to add new grouping variables.

If your existing file contains the variables SUBNUM, DATE, ENERGY, and FAT and you wish to include CHOLESTEROL and IRON, you only need to define the variables for CHOLESTEROL and IRON. Follow the same method of variable creation as explained in the section on entering data into a new file. When all of the new variables have been defined, just press <RETURN> when you are asked for the next variable.

You will now have a chance to change and rearrange the variables that you have just defined. The process is the same as entering data into a new file, so refer to that section if you need to. You cannot edit or alter the variables that already exist within the data file, only the ones that you have just created.

When all of the variables are in the correct order, jot down the variable label that is in the first slot. You will need this label name if you want to quit and resume entering data at a later date.

To begin entering data choose option 5. You will be asked to assign a value for missing numbers. This missing number value will be assigned to all records that have not been completed if you exit the program without entering all of the data for the new variables. This missing value can be changed once you resume entering. The missing value is assigned to unentered values so that each record will be in the same format.

Here is where the grouping variables come into play. You will be asked to enter the values for the grouping variables for each record. Using the example above, you would be asked to supply the subject number and date for each record you want to update. If you are going to enter the CHOLESTEROL and IRON values for subject 1 on date 12, enter 1 for SUBNUM and 12 for DATE. The program searches the file for the record that corresponds to that subject and date, and then will prompt you for the CHOLESTEROL and IRON values.

This part of the data entry is the same as before: enter each value followed by a <RETURN>. After entering all the new values for that record, you will be given the chance to view the values and change them if there are any mistakes. If you choose to check your data, you will be shown all the values for that record, not just the ones you have just entered, so you can change any of the values for that record, even the ones that already exist within the data file.

If you enter a grouping variable that does not correspond to a record in the file, you will be informed of an error. If you enter a combination that you have already updated, you will be told that you have already updated that record.

If you have finished entering data or wish to stop, just press <RETURN> when you are prompted for any of the grouping values. If you want to continue entering more data, choose Y in answer to the question. Otherwise choose N to exit the data entry section and return to the Main Menu.

When you choose to exit you will see two messages: "Checking Tree..." and "Saving Tree...." The program checks to see if there are any records that you did not update in that time. The variables that have not been assigned values will be assigned the missing number value by the program and information about the records that need to be updated is saved. If all the records have been updated, no information is saved.

This information about records that need to be updated will be lost if you choose to create another new variable for a data file that is in the process of being updated. If you choose to continue creating new variables, the information on which records still have to be updated will be erased and any record that has not been updated will retain the missing number value. The only way to change the missing values will be to choose the Edit A Record option on the Main Menu and edit each individual record.

If after exiting, you want to continue entering values into new variables, choose option 2, Continue Entering Values Into New Variables. You will be asked for the label of the variable that begins the list of new variables. This is the label that you should have jotted down when you first created the new variables. The program needs to know where the old variables end and the new variables begin.

Once you enter the variable label, the data entry process will begin. You will be asked for the missing number value, and everything will proceed as before.

After a session of entering new variables, a file called Filename.TRE will be created in the directory. This file contains information about which records have and have not been updated. This file is needed if you intend to continue entering data into the new variables. Once you have entered all the data, this file can be deleted. If the file is deleted before you have finished entering all of the data you will have to finish entering data by editing the record in that file.

4.6.6 Rearrange The Variables in The Data File

If you decide that you want the data stored in a different order, this option allows you to rearrange the data.

After choosing option 6 on the main menu and indicating that you want to continue, a sub-menu will be displayed with four options:

(1) Reorder One Variable - This option allows you to take one variable and place it at another spot in the list. For example, if your data list is:

SUBNUM DATE GROUP SODIUM POTASSIUM CALCIUM

and you want it to be:

SUBNUM DATE GROUP CALCIUM SODIUM POTASSIUM

you would choose to reorder one variable. When the program asks for a variable, you would enter 6 (CALCIUM is the sixth variable). When you are asked for the new position, enter 4 (you want to place calcium in the fourth position). CALCIUM now is in the fourth position, SODIUM in the fifth, and POTASSIUM in the sixth place.

(2) Exchange Two Variables - This option allows you to switch two variables in the data file. When asked for the first variable, enter the number of that variable. Do the same for the second variable. The two variables will be exchanged, with all others remaining the same.

(3) Finished, Rearrange File - When the variable list is in the order that you wish, this option will perform the changes on your data file. The data will be rearranged to fit your variable list. A new Filename.TXT file will list each variable and the columns they now reside in. The filename.FMT file will be updated, and when you enter more data into the file, the variables will be prompted in the new order that you defined. NOTE - once the file has been rearranged, the file name.DFT is erased. This means any customized headings that were defined for the report will be lost and will have to be reentered.

(4) Leave Without Rearranging - If after reordering the variable list you decide not to reorder the data file, this option will allow you to return to the Main Menu without any changes.

4.6.7 Change the Size of Numeric Variables

It is conceivable that you may have misjudged the size of a numeric variable and you now need to enter a value greater than the maximum allowed by the program. Another possibility is that you realize the maximum size of the variable is less than you had anticipated. By reducing the variable size you can impose a check on the field, insuring that no unnaturally large values are entered by mistake.

Choosing option 7 on the Main Menu allows you to enlarge or reduce the size of any of the numeric variables in the data file. After choosing this option, a sub-menu will ask you to verify that you really wish to change a variable's size. If not, choose sub-option 2, which will bring you back to the Main Menu; otherwise, enter 1 to continue with altering the size of numeric variables.

Upon choosing sub-option 1, you will be asked to enter the name of the new file. The file name must be the name of a file that exists and was created by this program. Remember to omit the dot extension.

Next, a variable list will be displayed. An example of such a listing is as follows:

```
1) SUBJECTI 3        2) DATE    C  6
3) WEIGHT  R  5 : 2  4) HEIGHT  R  5 : 2
```

You are now given two options. You may choose 1 to change the size of a numeric variable or 2 to finish and fix the data file. If you wish to change the size of a numeric variable, choose option 1 and you will be prompted to enter the number of the variable you wish to change. Note that if you choose a character variable you will receive a rude message disallowing the choice, so try to get it right the first time. If you choose an integer variable you will be asked to enter the number of digits in the largest value. If you choose a real one, you will be prompted for the length of the largest number (including the decimal point) and the number of digits after the decimal. Once you have made all of your changes, or decide not to make any, option 2 will process and change all of the records in the data file.

CHAPTER 5.0

FOOD CONSUMPTION ANALYSIS

Contributing Authors:
Carlo Radovsky
John Finn

5.1 Guideline for Consumption Data Analysis

This packet contains instructions for the processing and analysis of consumption data collected in the field, and the reasoning behind them. The task is broken down into the following sections:

5.1.1	Guidelines on Setup and Organization of the VAX
5.1.2	Useful Features and Programs on the VAX
5.1.3	File Transfer from the PC to the VAX
5.1.4	Cleanup, Merging, and Analysis of Consumption Data
5.1.5	Standard Statistical Procedures
5.1.6	Documentation of a Study's Directories

Appendix G: Program Listing and Technical Information

The standard abbreviations used in this packet are listed below:

<CR> - Indicates the Carriage Return/Enter key should be pressed.

<lower-case phrase> - Indicates a descriptive rather than an actual value and should be replaced by a value specific to the study.

Bold - Indicates a command to be typed at the appropriate prompt.

[lower-case phrase] - Indicates an optional parameter or keyword.

[,...] - Indicates multiple optional parameters similar to the one preceding this symbol.

CTRL - Control key should be depressed at the same time as the second key.

5.1.1 Setup and Organization of the VAX

5.1.1.1 General Information

Each study is allocated a set of directories, each directory being dedicated to a logical subset of the information relating to that study. All studies are contained within the Nutrition account and are actual sub-directories of the [NUTRITION] directory. This structure gives the information a "tree" organization that becomes more specific as you travel away from the "root" directory of Nutrition toward the outlying directories, the "leaves." Thus information is stored in a central location and accessible to all Nutrition computer users. This organization helps to maintain the integrity of the data, ensuring that the central copy of the information is always updated and accessed (see Sections 5.1.1.3 and 5.1.1.4 for more information concerning directories).

However, a drawback of this organization is that, since everyone accesses the central data, carelessness or ignorance can cause major setbacks. If many people are simultaneously using and transforming information, one could interfere with another's processes; a case of too many cooks spoiling the soup. Thus, be aware of other users and their needs. If you are experimenting with data, programs, or command files, be sure to safeguard the original, making sure it is not deleted or, even worse, confused or substituted with your working copy.

Although you may have files in many of the Nutrition directories, each one is traceable to you and accessible to everyone in Nutrition, unless you set protection. If someone else accidentally alters your data, it may take a while before the error is noticed. However, there is a daily backup of all files so that you cannot lose more than a day's worth of progress. Also note that when you create or edit a file, you become the current owner and the file uses your account's space quota.

Each account is limited in how much space it has available for storage of files. The limit is set by the Information Management Branch and is initially 5,000 blocks. In general you should own any programs or command files that you create or edit. If you are working with large or numerous data files it is best to have the Nutrition account own them. If you receive an error message stating that you have exceeded your account's space quota, contact the Information Management Branch or continue your work using the Nutrition account.

If you are at a crucial stage of development in the files that you are working on, you might wish to set protection on those files. But be cautious when setting protection. Be aware of who needs access to your files and do not lock them out. Also, please remember that exiting from a file will cause you to become the owner, so be cautious when exploring. To find the current protection of a file, type **DIR/PROTECTION <filename>**<CR>. If you receive a message stating that you do not have the privilege to access the file, then someone has protected that file. Since changing the protection of a file can be very complicated, see section 5.1.2.1.5 for more information.

Lastly, be aware of the fact that, when you log on, you are in the root directory of the account you are accessing. You must then move to the appropriate directory before beginning work.

For information on specific VAX commands and utilities see Section 5.1.2.

5.1.1.2 Command Usage on the VAX

Commands on the VAX are quite extensive and, as a result, somewhat complex and daunting until learned. There are some general guidelines that may help you.

Probably the foremost item to remember is that there are several help utilities you can consult. On-line help, system manuals, and a resident expert are the most common support avenues, hopefully relied upon in that order. The on-line help has nearly everything you will need and is fairly easy to use. It is organized in a similar fashion to the directory tree-structure in that you work your way down from general to specific information. Most software packages on the VAX can be learned fairly easily using manuals, although some of the more advanced features would best be learned from the resident expert.

Other items of importance are the VAX commands and their abbreviations. All standard VAX commands may be abbreviated to the point in which they are still a unique command. For example, the command words "SET DEFAULT" may be shortened to "SET DEF" but not to "S DEF" because there is a command "SHOW" which could also be represented as "S". Also be aware of the fact that the VAX is generally case insensitive; it does not differentiate between upper or lower case letters. Thus, "SET DEF" is equivalent to "set def" or "Set Def".

If, while typing a command, you find that it will not fit on one line, you can *make use of the* VAX *continuation character*. By placing a dash,"-", at the end of a line, you indicate to the computer that you wish to continue the command on the next line. When you hit <CR> the prompt will appear with an underline before it. This indicates a continuation. Type the rest of your command as usual, pressing <CR> when finished. The VAX allows a command to continue over several lines.

Lastly, there is a list of symbol commands that apply only to the Nutrition Division's computer users. These symbols are shortcuts, allowing you to execute common commands by typing short abbreviations in their stead. To examine the list type **TYL** at the prompt. This will scroll the list on the screen. To update or search the list for a specific symbol, type **EDL**. This will allow you to edit the file containing the symbols. Be sure that any additions you make are inserted in alphabetical order and are unique to both the list of symbols and VAX commands (i.e. that they are not redefining symbols previously used to perform some other function). If you are not making any additions be sure to quit from the file. To get an idea of the usefulness of these symbols, both TYL and EDL are symbols, representing "TYPE DISK$USER2:[NUTRITION]SYMBOLS.COM" and "EDIT DISK$USER2:[NUTRITION]SYMBOLS.COM" respectively. If you have a general idea of a symbol command in the list but cannot remember it exactly, there is a way to display only those items that fit the general form you want. Type **SHOW SYMBOL** followed by some approximation of the symbol you want, set off by the wild card character, "*". To show all those symbols that begin with the letter "B", you would type the following: **SHOW SYMBOL B*** <CR>. To show all those symbols containing a "B", type **SHOW SYMBOL *B***<CR>.

5.1.1.3 Creating a Directory

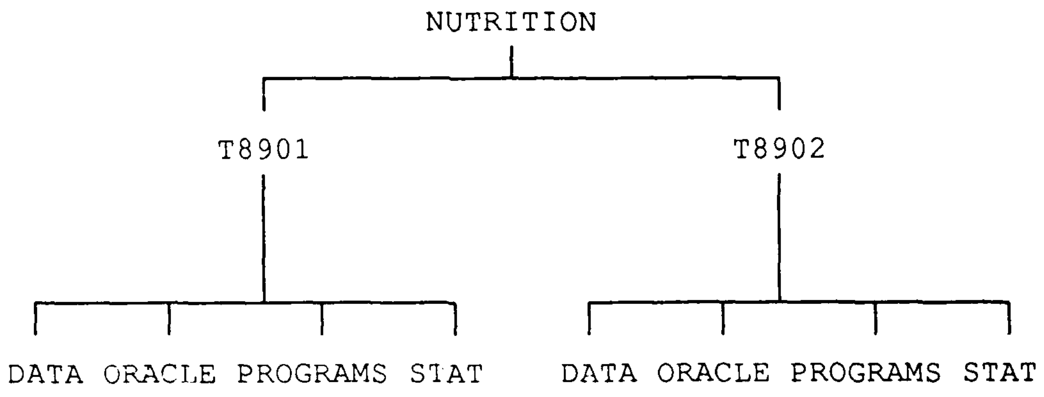
The VAX command for creating a directory is **CREATE/DIRECTORY** followed by the directory path, enclosed in brackets. Note that it is possible to create a series of nested sub-directories with a single **CREATE/DIRECTORY** command. For example, [NUTRITION.T9001.DATA.TEMP] can be created, even though neither [NUTRITION.T9001.DATA] nor [NUTRITION.T9001] exists at the time the command is issued. Each sub-directory will be created, starting with the highest level and proceeding downwards. No wild card characters are allowed in the directory specification. An example follows:

```
$CREATE/DIRECTORY [NUTRITION.T9001]<CR>
```

This command has two effects. The first is that it creates a sub-directory of [NUTRITION] named [NUTRITION.T9001]. The second effect is that a directory file named T9001.DIR is placed in the [NUTRITION] directory. This file contains information relating to the files stored in the [NUTRITION.T9001] directory.

The organization of directories, study directories in particular, should follow the standard in Figure 5.1. When a new set of data or programs is to be created, there should be a root directory which serves only as an address, so to speak, for the rest of the information. The naming convention for a study's main directory is a "T", followed by the year of the study, followed by the sequence in which the study *occurs in the year. Thus the second* study of 1990 would have a main directory named "T9002". The conventions for the sub-directories of a study can be seen in Figure 5.1. If the set of a study's directories is to be placed in the Nutrition directory, it should be directly below the root. A diagram is as follows:

Figure 5.1



Using the example above, the T8901 directory would contain no information except the sub-directory names. Related information is then stored in the appropriate sub-directory of T8901.

5.1.1.4 Methods of Moving Around the VAX

There are two main ways of moving around the directory tree: the standard VAX commands and the Nutrition symbols set up in a central table (Section 5.1.1.2). The VAX command to change from one directory to another is **SET *DEFAULT* [<*directory* path>]**, where the "directory path" consists of the root directory followed by the specific sub-directories desired, each separated by a period (see example below). The symbol "CD" - Change Directory - is defined for Nutrition users which may be used in place of the words "SET DEFAULT". The following three commands are identical in performance; they would take the user to the statistics directory of the Ft. Hood study:

```
$SET DEFAULT [NUTRITION.T8804.STAT]<CR>
$CD [NUTRITION.T8804.STAT]<CR>
$HOODSTAT<CR>
```

If your current location is somewhere along a directory path, you may omit those elements of the path that define your current location. For example, if you are positioned in the [NUTRITION.T8804] directory, you could move to that study's statistics directory by typing **CD [.STAT]**<CR>. Note that the first character of the path is a period; this indicates that you are moving to a *sub-directory of your* current location and not to an entirely different directory. Also realize that you may include several sub-directories in the path, thus moving to a remote sub-directory of your current location.

Another feature is the occurrence of a dash in the directory path which will cause you to move "up" one directory in the tree structure. Thus, to move from the directory [NUTRITION.T8804.STAT] to [NUTRITION.T8804], you could type **CD [-]**<CR>, whereas to move from [NUTRITION.T8804.STAT] to the [NUTRITION.T8804.DATA] directory you could type **CD [-.DATA]**<CR>. Also note that you can move up several directories at a time by placing sequential dashes in the directory path. Thus, typing **CD [--]**<CR> would take you from [NUTRITION.T8804.DATA] to the [NUTRITION] directory.

5.1.2 Useful Features and Programs on the VAX

5.1.2.1 VAX Features

There are several features on the VAX that come in handy in the processing of data. The most commonly used ones are discussed in detail in this section, each generally broken down into five sections: common uses, format, parameters, command_qualifiers, and examples. Although each section is fairly comprehensive, this documentation could by no means hold all relevant information concerning the features of the VAX. See section 5.1.2.1.1 to find out where you can obtain more information.

The following is a brief overview of each feature:

5.1.2.1.1 Use of the Help Utility - Describes how to access the on-line help.

5.1.2.1.2 Directory - Provides a list of files or information about a file or group of files.

5.1.2.1.3 Search - The VAX SEARCH command searches one or more files for the specified string or strings and lists all the lines containing occurrences of the strings.

5.1.2.1.4 Differences - Compares the contents of two disk files and creates a listing of those records that do not match.

5.1.2.1.5 Set Protection - Establishes the protection to be applied to a particular file or a group of files. The protection of a file limits the type of access available to system users

5.1.2.1.6 Sort - Invokes the VAX Sort Utility (SORT) to reorder the records in a file into a defined sequence and to create either a new file of the reordered records or an address file by which the reordered records can be accessed

5.1.2.1.7 Rename - Changes the directory specification, file name, file extension, or file version of an existing disk file or disk directory.

5.1.2.1.8 Copy - Creates a new file from one or more existing files while leaving the old file intact.

5.1.2.1.9 Append - Adds the contents of one or more specified input files to the end of the specified output file.

Any feature that you start running can be interrupted. There are two ways to do this. First, pressing **CTRL-Y** will stop the entire process and return you to the DCL prompt. This is a very powerful abort which completely stops all processes. The second way, pressing **CTRL-C**, is less powerful in that it aborts only the current function of a process and jumps to the next function. A good example of this is the **TYPE** command. If you are typing many files to the screen (i.e. **TYPE** ***.DAT**<CR>), and are halfway through the third file when you realize that it is not the one you are looking for, typing **CTRL-C** will stop the listing of the third file and immediately start scrolling the fourth file. Typing **CTRL-Y**, on the other hand, will completely stop the scrolling of the files and return you to the DCL prompt.

Several of the features of the VAX have command qualifiers that allow you to specify a time with which to differentiate between information. The following is a description of the time command qualifier.

Format

DD-MMM-YYYY:HH:MM:SS.CC

Parameters

• DD	-	A two digit number referring to the day of the month.
• MMM	-	A three letter code for the month (i.e. JAN, FEB, MAR, etc.)
• YYYY	-	A four digit number referring to the year (i.e. 1988)
• HH	-	A two digit number referring to the hour (note that it is a 24 hour clock)
• MM	-	A two digit number referring to the minutes
• SS	-	A two digit number referring to the seconds
• CC	-	A two digit number referring to the hundredths of seconds

Note that the dashes, colons, and the period are required characters, and if any of these parameters are left out of the time specification, the current values will be assumed. Specific parameters in the below features may also allow key words for the time.

The features discussed in this section have been organized into two logical groups. The first three features (Sections 5.1.2.1.2 - 5.1.2.1.4) are descriptive in that they do not change files in any way but simply report information. The rest of the features (5.1.2.1.5 - 5.1.2.1.9) help the user to manipulate data and files.

Lastly, keep in mind the conventions noted at the beginning of this document.

5.1.2.1.1 Use of the Help Utility

Although the following discussions are quite detailed, you should be aware of the fact that a more in-depth explanation of each feature may be found either in the on-line help utility or in the VAX/VMS Utility Reference Manuals. The VAX/VMS Utility Reference Manuals are held by the Information Management Branch and can be loaned out.

The HELP utility displays information about requested VMS topics. From the DCL command level (in response to the $ prompt), you can display a list of topics for which help information is available by typing **HELP** and pressing the <CR> key. The system responds by displaying a brief description of how to use HELP, followed by a list of topics for which help is available, followed by the prompt "Topic?". You can exit from the HELP facility by typing **CTRL-Z** (that is, pressing the CONTROL and Z keys simultaneously) in response to any prompt.

To display information on a particular topic, respond to the prompt by typing the name of the topic and pressing the <CR> key. If you are not sure of the name of the topic for which you need help, type the name HINTS. To display information on all available topics, type an asterisk (*). To display all the information on a topic, type the topic name immediately followed by an ellipsis (...), e.g., **SHOW...**

You can specify percent signs (%) and asterisks (*) in the keyword as wild card (i.e., match all) characters. The percent sign is a wild card for a single character (AB% could be ABC, ABD, ABA, etc.), while the asterisk is a wild card for any amount of characters (AB* could be ABJDJDJD, ABC, ABDHDH, etc.) Abbreviations result in all matches being displayed. *Note that the wild card characters and the ellipsis are applicable to many other features of the VAX.*

The information displayed by HELP on a particular topic includes a description of the topic and a list of subtopics that further describe the topic. To display subtopic information, type one of the subtopic names from the list in response to the "Subtopic?" prompt.

If you press <CR> in response to the "Subtopic?" prompt instead of typing a subtopic name, the "Topic?" prompt reappears, enabling you to enter another topic name. If you press RETURN in response to the "Topic?" prompt, you will exit from HELP.

You can type a question mark (?) in response to any of the prompts to re-display the most recently requested text and a list of topic or subtopic names. For example, if you type ? in response to the "Subtopic?" prompt, a list of subtopics is displayed followed by the "Subtopic?" prompt.

5.1.2.1.2 Directory

Provides a list of files or information about a file or group of files. Helps to locate files as well as describe their attributes.

Format

```
DIRECTORY  [file-spec[,...]]
```

Parameters

file-spec[,...]

Specifies one or more files to be listed. The syntax of a file specification determines which files will be listed, as follows:

- If you do not enter a file specification, the DIRECTORY command lists all versions of the files in the current default directory.
- Whenever the file specification does not include a file name and file extension, all versions of all files in the specified directory are listed.
- If a file specification contains a file name and/or file extension and no version number, the DIRECTORY command lists all versions.
- If a file specification contains only a file name, the DIRECTORY command lists all files in the current default directory with that name, regardless of file extension and version number.
- If a file specification contains only a file extension, the DIRECTORY command lists all files in the current default directory with that file extension, regardless of file name and version number.
- If a file specification contains an ellipsis (...) within the brackets of the directory specification, you are referring to the current directory along with all of its sub-directories. Note that this applies to all directory specifications along wlth the dash and wild cards characters.

If you specify more than one file, separate the file specifications with either commas or plus signs. You can use wild card characters in the directory specification, file name, file extension, or version number fields of a file specification to list all files that satisfy the components you specify.

Command Qualifiers

- /BEFORE[=time]

Selects only those files that are dated before the specified time. You can specify either an absolute time or a combination of absolute and delta times. See Section 5.1.2.1.1 for general information concerning specifying time values or Section 2.5 of the VAX/VMS DCL Concepts Manual for complete information on specifying time values. You can also use the keywords TODAY, TOMORROW, and YESTERDAY. If no time is specified, TODAY is assumed.

- /SINCE[=time]

Selects only those files that are dated after the specified time. You can specify either an absolute time or a combination of absolute and delta times. You can also use the keywords TODAY, TOMORROW, and YESTERDAY. If no time is specified, TODAY is assumed.

- /DATE[=option]
  /NODATE

Includes the backup, creation, expiration, or modification date for each specified file. If you omit this qualifier, the default is /NODATE. If you use the /DATE qualifier without an option, the creation date is provided. You can specify one of the following options with the /DATE qualifier:

ALL	Lists all four file dates, in this order: CREATED, MODIFIED, EXPIRED, BACKUP.
BACKUP	Lists the date of the last backup with each file.
CREATED	Lists the creation date with each file.
EXPIRED	Lists the expiration date with each file.
MODIFIED	Lists the last date the file was written.

- /PROTECTION
  /NOPROTECTION (default)

Controls whether the file protection for each file is listed.

- /OWNER
  /NOOWNER (default)

Controls whether the file owner's User Identification Code (UIC) is listed. The default size of the owner field is 20 characters. If the file owner's UIC exceeds the length of the owner field, the information will be truncated. The size of this field can be altered by specifying /WIDTH=OWNER, along with a value for the OWNER field. For more information, see the description of the /WIDTH qualifier.

- /BY_OWNER[=UIC]

Selects one or more files only if their owner UIC matches the specified owner UIC. Specify the UIC using standard UIC format as described in Section 7.1.1 of the VAX/VMS DCL Concepts Manual. If the /BY_OWNER qualifier is specified without a UIC, the UIC of the current process is assumed.

- /SIZE[=option]
  /NOSIZE (default)

Provides the file size in blocks used and/or allocated for each file listed, according to the option you specify. If you specify /SIZE without an option, the listing provides the file size in blocks used. The options you can specify are:

ALL	Lists the file size both in blocks used and blocks allocated
ALLOCATION	Lists the file size in blocks allocated
USED	Lists the file size in blocks used

The size of this field can be altered by supplying the SIZE value of the /WIDTH qualifier.

- /EXCLUDE=(file-spec[,...])

Any files that match the listed file specifications are excluded from the DIRECTORY operation. If you specify only one file, you can omit the parentheses. Wild card characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version. The file specification can contain a directory specification, but not a device specification.

- /OUTPUT[=file-spec]
  /NOOUTPUT

Controls where the output of the command is sent. If you do not enter the qualifier, or if you enter /OUTPUT without a file specification, the output is sent to the current process default output stream or device, identified by the logical name SYS$OUTPUT.

If you enter /OUTPUT with a partial file specification (for example,

/OUTPUT=[JONES]), DIRECTORY is the default file name and LIS the default file extension. If you enter a file specification, it may not include any wild card characters.

If you enter /NOOUTPUT, output is suppressed.

- /FULL

Lists the following items for each file:

File name	File extension
Version number	Number of blocks used
Number of blocks allocated	Date of creation
Date of last backup	Date last modified
Date of expiration	File owner's UIC
File protection	File identification number (FID)
File organization	Other file attributes
Record attributes	Record format
Access control list (ACL)	

The /FULL qualifier overrides the default brief listing format.

- /TOTAL

Suppresses the listing of all individual file information and displays only the trailing lines, as described under the /TRAILING qualifier. By default, the output format is /BRIEF, which gives this total, but also lists all the file names, file extensions, and their version numbers.

- /GRAND_TOTAL

Suppresses both the per-directory total and individual file information. /GRAND_TOTAL displays only the total line for all files and directories that have been selected. (See the /TRAILING qualifier for information on displaying directory totals.)

- /TRAILING
  /NOTRAILING

Controls whether tr ing lines that summarize the following information are output:

- Number of files listed
- Total number of blocks used per directory
- Total number of blocks allocated

- Total number of directories and total blocks used and/or allocated in all directories (only if more than one directory is listed)

By default, the output format includes most of this summary information. The /SIZE and /FULL qualifiers determine more precisely what summary information is included. If you omit /SIZE or /FULL, only the number of files is displayed and possibly the total number of directories, if applicable. If you specify the /SIZE qualifier, the number of blocks is also displayed, according to the size option selected (USED and/or ALLOCATION). If you specify the /FULL qualifier, the number of files as well as the number of blocks used and allocated are displayed.

<u>Examples</u>

1. **$DIRECTORY**<CR>

The DIRECTORY command lists all versions of all files in the current default disk and directory in the brief format. The heading identifies the disk and directory, and the trailing line gives the total number of files.

2. **$DIRECTORY BLOCK%%%**<CR>

The DIRECTORY command locates all versions and extensions of files in the default device and directory whose names begin with the letters BLOCK and end with any three additional characters. The default output format is brief, four columns, with heading and trailing lines.

3. **$DIRECTORY/BEFORE=01-JAN-1988**<CR>

The DIRECTORY command lists only those files that were updated before January 1, 1988. Thus if you know that the file you are looking for was last updated in February of 1988, you could list all files of that date by using both the /BEFORE and /SINCE qualifiers to isolate them.

4. **$DIRECTORY/EXCLUDE=(*.DAT,*.PLT)**<CR>

The DIRECTORY command lists all the files catalogued in the default directory <u>except</u> those with the file extensions of DAT and PLT.

5. **$DIRECTORY/OUTPUT=LISTING.OUT**<CR>

The DIRECTORY command places the list of all files in the default directory into the file LISTING.OUT. This file can be printed from the DCL prompt or examined in the editor.

6. **$DIRECTORY [NUTRITION.T8901...]**<CR>

The DIRECTORY command lists all of the files cataloged in the [T8901] directory along with all of the files in all of the sub-directories of [T8901].

5.1.2.1.3 Search

The VAX SEARCH command searches one or more files for the specified string or strings and lists all the lines containing occurrences of the strings. This command is useful in locating a file when you do not know its name but can come up with a fairly unique string contained within the file.

Format

SEARCH file-spec[,...] search-string[,...]

Parameters

- file-spec[,...]

Specifies the names of one or more files to be searched. You must specify at least one file name. If you specify two or more file names, separate them with commas. Wild card characters are allowed in the file specification.

- search-string[,...]

Specifies one or more strings to search for in the specified files. If the search string contains any lowercase letters or non-alphanumeric characters (including spaces), enclose it in quotation marks. You can use the /EXACT qualifier to alter the way that SEARCH matches search strings.

Command Qualifiers

- /WINDOW[=(n1,n2)]
  /NOWINDOW (default)

Controls the number of lines that are listed along with the line containing the matching string. If you specify the /WINDOW qualifier with a single number n, n-1 additional lines are displayed with each line containing the search string. Half of the additional lines are listed above the line containing the match, and half are listed below. If n-1 is an odd number, the extra line is listed below the search string. For example, if you specify /WINDOW=10, nine additional lines are listed along with the line containing the search string, four lines are listed above the line containing the search string and five lines are listed below it, making a total of ten lines.

If you specify /WINDOW without specifying a number, the default number of five lines---two above, one containing the search string, and two below---is used.

If the form /WINDOW=(n1,n2) is used, n1 refers to the number of lines above the matched line and n2 refers to the number of lines below. Either of these numbers can be zero.

If /WINDOW=0 is specified, SEARCH will display the file name of each file containing a match, but no records. You can use this specification to create a file (using the /OUTPUT qualifier) that can be inserted into a command file to manipulate the files containing matches.

If you omit the /WINDOW qualifier entirely, only the line in which the match is satisfied is displayed.

- /EXACT
  /NOEXACT (default)

Controls whether the SEARCH command matches the search string exactly, or treats uppercase and lowercase letters as equivalents. The default qualifier, /NOEXACT, causes SEARCH to ignore case differences in letter.

Specifying the /EXACT qualifier causes the system to use less CPU time. Therefore, if you are sure of the case of the letters in the string, it is more efficient to use /EXACT.

- /EXCLUDE=(file-spec[,...])

Causes the SEARCH command to exclude the listed file specifications from the search. Do not include a directory or device name in this file specification. Wild card characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version. If you use this qualifier, you must include at least one file specification. If you specify only one file, you can omit the parentheses. If you omit the /EXCLUDE qualifier, SEARCH will examine all files in the input file list.

- /OUTPUT[=file-spec]
  /NOOUTPUT

Controls whether the results of the search are output to a specified file. The output will be sent to the current default output device (SYS$OUTPUT) if you omit the /OUTPUT qualifier or omit the file specification with the qualifier.

The /NOOUTPUT qualifier means that no matching records are output as a result of the SEARCH command.

Examples

1. **$SEARCH CABLE.MEM,JOYNER.MEM "MANUAL TITLE"**<CR>

This command searches the files CABLE.MEM and JOYNER.MEM for occurrences of the character string MANUAL TITLE. Each line containing the string is displayed at the terminal. It is necessary to enclose the string in quotation marks because it contains a space character.

2. **$SEARCH/OUTPUT=RESULTS.DAT/WINDOW=9 -**<CR>
   _$DISLIST.MEM NAME<CR>

The SEARCH command searches the file DISLIST.MEM for occurrences of the character string NAME and sends the output to the file RESULTS.DAT. The four lines preceding and following each occurrence of NAME are included in the output.

5.1.2.1.4 Differences

Compares the contents of two disk files and creates a listing of those records that do not match. This command allows you to identify changes between two files or determine the similarity between two files of different names.

Format

DIFFERENCES master-file-spec [revision-file-spec]

Parameters

- master-file-spec

Specifies the name of the primary input file to be compared. The file specification must include a file name and a file extension. No wild card characters are allowed in the file specification.

- revision-file-spec

Specifies the name of the secondary input file to be compared. Any unspecified fields default to the corresponding fields of the primary input file specification. If you do not specify a secondary input file, the DIFFERENCES command uses the next lower version of the primary input file.

No wild card characters are allowed in the file specification.

Examples

1. **$DIFFERENCES EXAMPLE.TXT**<CR>

```
************
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
        1  DEMONSTRATION
        2  OF V3.0 DIFFERENCES
        3  UTILITY
******
File DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
        1  DEMONSTRATION
        2  OF VMS DIFFERENCES
        3  UTILITY
************

Number of difference sections found: 1
Number of difference records found: 2

DIFFERENCES/MERGED=1-
      DISK1:[GEORGE.TEXT]EXAMPLE.TXT;2
      DISK1:[GEORGE.TEXT]EXAMPLE.TXT;1
```

The DIFFERENCES command compares the contents of the two most recent versions of *the file* EXAMPLE.TXT *in the current default directory*. DIFFERENCES compares every character in every record and displays the results at the terminal.

2. **$DIFFERENCES EXAMPLE.TXT NEWEXAMP.TXT**<CR>

The DIFFERENCES command compares the contents of the file EXAMPLE.TXT to the contents of the file NEWEXAMP.TXT in the current default directory. DIFFERENCES compares every character in every record and displays the results at the terminal.

5.1.2.1.5 Set Protection

Establishes the protection to be applied to a particular file or a group of files. the protection of a file limits the type of access available to system users. This is a feature that should be used with great discretion. Once you set protection on a file, all subsequent versions of the file will have that same protection access. If you set protection on a file and forget to remove it later on, difficulties can arise that are hard to trace.

Format

```
SET PROTECTION[=(code)] file-spec[,...]
```

Parameters

- code

Defines the protection to be applied to the specified files. If no code is included, the access of the specified files is set to the current default protection. The format for the code parameter is described in detail in Section 7.1.2 of the VAX/VMS DCL Concepts Manual. The following is a brief discussion of the qualifiers of the code parameter.

You may set protection on four access categories; System, Owner, Group, and World.

- The System category refers to the operator account and any functions that are performed in relation to system maintenance. Make sure that you always allow the system full access to all of your files; if the system cannot access your files, it cannot back them up.

- The Owner category refers to the account of the owner of the file. In general you want to give full access to this group. An exception to this rule occurs when you wish to make sure that all of the past versions of a file are saved and cannot be deleted. In order to protect the file against deletion by you (the owner), you also need to protect the file against deletion by all other access categories. The following command shows the proper way to do this.

  $SET PROTECTION= -<CR>
  _$(OWNER:RWE,GROUP:RWE,WORLD:RWE) INCOME.DAT;1<CR>

- The Group category refers to all users in your access group which, in this case, are all Nutrition users. You will want to give full access to the group unless you wish to stop anyone else from tampering with your files.

- The World category refers to all VAX users. Unless there is a specific reason you wish to give access to someone outside of your group, keep this access restricted.

For each of the access categories there are four access modes: Read, Write, Execute, and Delete.

- The Read mode allows the file to be edited.

- The Write mode allows the file to be copied.
- The Execute allows the file to be run.
- The Delete mode allows the file to be deleted or purged.
- By omitting all of these modes from a category, you will be allowing no access to that category.

See the examples below for uses of the code.

- file-spec[,...]

Specifies one or more files for which the protection is to be changed. A file name and file extension are required; if you omit a version number, the protection is changed for only the highest existing version of the file. You can specify wild card characters in the directory, file name, file extension, and version fields.

<u>Command Qualifiers</u>

- /CONFIRM
  /NOCONFIRM (default)

Controls whether the SET PROTECTION command displays the file specification of each file before applying the new protection, and requests you to confirm that the file's protection should be changed. If you specify /CONFIRM, you must respond to the prompt with a Y (YES) or a T (TRUE), and then press RETURN before the SET PROTECTION command will change the file protection. If you enter anything else, such as N or NO, the requested file protection is not applied.

- /LOG
  /NOLOG (default)

Controls whether the SET PROTECTION command displays the file specification of each file after it has reset the protection.

- /PROTECTION=(code) File-spec qualifier.

Defines the protection code to be applied to the associated file specification. Use this qualifier to assign different protection codes to several files with a single SET PROTECTION command.

If you specify the command's code parameter in addition to using the /PROTECTION qualifier with a file specification, the attributes specified with the command's code parameter are applied first. Any attributes specified with the /PROTECTION qualifier override the command's code parameter attributes.

Specify the protection code using the format described in Section 7.1.2 of the VAX/VMS DCL Concepts Manual.

Examples

1.
```
$SET PROTECTION -<CR>
_$PAY.LIS/PROTECTION= -<CR>
_$(SYSTEM:R,OWNER:RWED,GROUP:RW),-<CR>
_$PAY.OUT/PROTECTION=(SYSTEM:RWED,GROUP:RWED,W)<CR>
```

The SET PROTECTION command changes the protection codes applied to two files. To the file PAY.LIS, it gives the system read-only access; the owner is given read, write, execute, and delete access; and users in the owner's group are given read and write access. To the file PAY.OUT, it gives the system and group all types of access; the current access for the owner does not change, but the world is denied all types of access.

2.
```
$SET PROTECTION A.DAT, B.DAT -<CR>
_$/PROTECTION=OWNER:RWED, C.DAT<CR>
```

The SET PROTECTION command specifies that the file A.DAT receive the default protection established for your files. The existing protection for the file B.DAT is overridden, only for the owner category, to provide read, write, execute, and delete access. Note that no protection is specified for the file C.DAT at either the command or file level. Thus, like A.DAT, C.DAT receives the default protection. Since no version numbers are specified, the protection settings affect only the highest versions of the three files.

3.
```
$SET PROTECTION=OWNER:D -<CR>
_$[MALCOLM.SUB1]SUB2.DIR/PROTECTION=GROUP:D<CR>
```

The SET PROTECTION command changes the protection for the owner and group categories of the sub-directory [MALCOLM.SUB1.SUB2] to permit deletion. However, the protection for the world and system categories is not changed.

4.
```
$DIR/PROTECTION INCOME.DAT<CR>

    Directory DBA0:[SMITH]

    INCOME.DAT;2          (RWED,RWED,RWED,RWED)
    INCOME.DAT;1          (RWED,RWED,RWED,RWED)

    Total of 2 files.

$SET PROTECTION=(OWNER:RWE) INCOME.DAT;1<CR>
$PURGE<CR>
```

The file INCOME.DAT;1 has been protected against deletion by the owner. However, since the owner is also a member of the group and world categories, the file is still vulnerable to deletion. The subsequent PURGE command will delete INCOME.DAT;1.

5.1.2.1.6 Sort

Invokes the VAX Sort Utility (SORT) to reorder the records in a file into a defined sequence and to create either a new file of the reordered records or an address file by which the reordered records can be accessed. For a complete functional description of the Sort Utility, including more information about the SORT command, see the VAX/VMS Sort/Merge Utility Reference Manual.

Format

```
SORT input-file-spec[,...] output-file-spec
```

Parameters

- input-file-spec[,...]

Specifies the name of the files whose records are to be sorted. You can sort up to 10 input files to create one output file. Multiple input file specifications must be separated by commas. If the file specifications do not include a file extension, SORT assumes the default file extension of DAT. No wild card characters are allowed in the file specification.

- output-file-spec

Specifies the name of the file into which the sorted records are to be written. If an address sort or index sort is selected, output file specification names the address file. If the file specification does not include a file extension, SORT uses the file extension of the first input file. No wild card characters are allowed in the file specification.

Command Qualifiers

- /KEY=(field[,...])

Defines a sort key, including position, size, order, and data type. If the key field starts in the first position, encompasses the entire record, will be sorted in ascending order, and contains character data, the /KEY qualifier need not be specified.

The key field is defined by keywords that must be separated with commas and enclosed in parentheses.

Required Keywords

POSITION:n	Specifies the starting position of the sort key field within each record, where the first character of each record is position 1.
SIZE:n	Specifies the length of the sort key field in characters, bytes, or digits, depending on the key field data type. The total of all key field sizes must be less than 32767 bytes. The valid sizes, based on data types, are:

Data Type	Values for n
Character	1 through 32767 bytes
Binary	1, 2, 4, 8, 16 bytes
Decimal	1 through 31 digits

Optional Keywords

NUMBER:n	Specifies the precedence of the sort key being defined, where 1 represents the primary sort key, 2 represents the secondary sort key, and so on. If this keyword is not specified in the first /KEY qualifier, NUMBER:1 is assumed; if not specified in any subsequent /KEY qualifiers, the default value is the number assigned to the previous key, plus 1. The legal values are 1 through 255.
CHARACTER	Indicates that character data appears in the sort key field. If no data type is specified, SORT assumes that the data type is CHARACTER.
DECIMAL	Indicates that decimal data appears in the sort key field. If no data type is specified, SORT assumes that the data type is CHARACTER.
ASCENDING	Indicates that the key is to be sorted into ascending order. The default is ASCENDING.
DESCENDING	Indicates that the key is to be sorted into descending order. The default is ASCENDING.

Examples

```
1. $SORT/KEY=(POSITION:1,SIZE:70) -<CR>
   _$BOATS.LST  BOATS.TMP<CR>
```

This SORT command sorts the records in the file BOATS.LST and creates an output file named BOATS.TMP. All the records in the input file are sorted in alphanumeric order based on the first 70 characters in each record.

2. **$SORT/KEY=(POSITION:47,SIZE:2)-<CR>**
   _$/KEY=(POSITION:51,SIZE:7)-<CR>
   _$BOATS.LST,BOATS2.LST BOATS.BEM<CR>

This SORT command specifies two key fields within the two input files, BOATS.LST and BOATS2.LST: the first key is in columns 47 and 48 of each input record, and the second key is in columns 51 through 57. The output file is named BOATS.BEM. All the records in the input files are sorted in alphanumeric order based first on the characters in columns 47 and 48, and then on the characters in columns 51 through 57.

5.1.2.1.7 Rename

Changes the directory specification, file name, file extension, or file version of an existing disk file or disk directory.

Format

```
RENAME  input-file-spec[,...] output-file-spec
```

Parameters

- input-file-spec[,...]

Specifies the names of one or more files whose specifications are to be changed.

You can use wild card characters in the directory specification, file name, file extension, or version number fields of the file specification. When wild card characters are used, all files whose specifications satisfy the wild card fields are renamed.

- output-file-spec

Provides the new file specification to be applied to the input file. The RENAME command uses the device, directory, file name, and file extension of the input file specification to provide defaults for fields in the output file that are either not specified, or indicated by a wild card character. Wild card characters in corresponding fields of the input and output file specification result in multiple rename operations.

The RENAME command supplies output file version numbers according to the first description below that applies:

1. If the output file specification contains an explicit version number, the RENAME command uses that version number.

2. If the input file specification or output file specification contains an asterisk in the version number field, the RENAME command uses the version number of each input file to name the corresponding output file.

3. If no file currently exists with the same file name and file extension as that specified for the output file, the RENAME command gives the new file a version number of 1.

4. If a file currently exists with the same file name and file extension as that specified for the output file, the RENAME command gives the output file a version number one greater than the highest existing version, unless the /NONEW_VERSION qualifier is specified.

Command Qualifiers

- /BEFORE[=time]

Selects only those files that are dated before the specified time.

You can specify either an absolute time or a combination of absolute and delta times. See Section 2.5 of the VAX/VMS DCL Concepts Manual for complete information on specifying time values. You can also use the keywords TODAY, TOMORROW, and YESTERDAY. If no time is specified, TODAY is assumed.

- /SINCE[=time]

Selects only those files that are dated after the specified time. You can specify either an absolute time or a combination of absolute and delta times. See Section 2.5 of the VAX/VMS DCL Concepts Manual for complete information on specifying time values. You can also use the keywords TODAY, TOMORROW, and YESTERDAY. If no time is specified, TODAY is assumed.

- /EXCLUDE=(file-spec[,...])

Any files that match the listed file specifications are excluded from the RENAME operation. If you specify only one file, you can omit the parentheses. The file specification can contain a directory specification. Wild card characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version. The file specifications cannot include a device name.

- /BY_OWNER[=UIC]

Selects one or more files only if their owner user identification code (UIC) matches the specified owner UIC.

Specify the UIC using standard UIC format as described in Section 7.1.1 of the VAX/VMS DCL Concepts Manual. If the /BY_OWNER qualifier is specified without a UIC, the UIC of the current process is assumed.

Examples

1. **$RENAME AVERAGE.OBJ OLDAVERAGE**<CR>

The RENAME command changes the file name of the highest existing version of the file AVERAGE.OBJ to OLDAVERAGE.OBJ. If no file named OLDAVERAGE.OBJ currently exists, the new file is assigned a version number of 1.

2. **$RENAME *.TXT;* *.OLD;***<CR>

The RENAME command renames all versions of all files with the file extension TXT to have the file extension OLD. The file names and version numbers are not changed.

5.1.2.1.8 Copy

Creates a new file from one or more existing files. The COPY command can:

- Copy an input file to an output file
- Join two or more input files into a single output file
- Copy a group of input files to a group of output files

Format

```
COPY  input-file-spec[,...] output-file-spec
```

Parameters

- input-file-spec[,...]

Specifies the names of one or more input files to be copied. If you specify more than one input file, you can separate the names with either commas or plus signs. You can use wild card characters in the file specifications.

- output-file-spec

Specifies the name of the output file into which the input files will be copied. You must specify at least one field in the output file specification. If the device or directory is not specified, your current default device and directory are used. The COPY command replaces any other missing fields (file name, file extension, version number) with the corresponding field of the input file specification. If you specify more than one input file, COPY generally uses the fields from the first input file to determine any missing fields in the output file.

The asterisk wild card character can be used in place of the file name, file extension, and/or version number. The COPY command uses the corresponding field in the related input file to name the output file. The wild card character can also be used in the output file specification to have COPY create more than one output file. For example:

```
$COPY A.A;1, B.B;1 *.C<CR>
```

This COPY command creates the files A.C;1 and B.C;1 in the current default directory. Full use of wild card characters is allowed for directories in the output file specification.

<u>Command Qualifiers</u>

- /EXCLUDE=(file-spec[,...])

Any files that match the listed file specifications are excluded from the COPY operation. If you specify only one file, you can omit the parentheses. Wild card characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version. The file specification can contain a directory specification, but not a device specification.

<u>Examples</u>

1. **$COPY TEST.DAT NEWTEST.DAT**<CR>

The COPY command copies the contents of the file TEST.DAT from the default disk and directory to a file named NEWTEST.DAT on the same disk and directory. If a file named NEWTEST.DAT already exists, the COPY command creates a new version of it.

2. **$COPY ALPHA.TXT TMP**<CR>
   $COPY ALPHA.TXT .TMP<CR>

The first COPY command copies the file ALPHA.TXT into a file named TMP.TXT. The COPY command uses the file extension of the input file to complete the file specification for the output file. The second COPY command creates a file named

ALPHA.TMP. The COPY command uses the file name of the input file to name the output file.

3. **$COPY/EXCLUDE=J*.DAT *.DAT *.TXT**<CR>

The COPY command copy all files with the file extension of DAT, except those beginning with J, to files with the same file name but with the file extension of TXT.

5.1.2.1.9 Append

Adds the contents of one or more specified input files to the end of the specified output file.

Format

```
APPEND  input-file-spec[,...] output-file-spec
```

Parameters

- input-file-spec[,...]

Specifies the names of one or more input files to be appended to the output file. If you specify more than one input file, separate the specifications with either commas *or plus signs (commas and plus signs are equivalent)*. All input files are appended, in the order specified, to the end of the output file.

You can use wild card characters in the input file specifications.

- output-file-spec

Specifies the name of the file to which the input files will be appended. You must include at least one field in the output file specification. If you do not specify a device and/or directory, the APPEND command uses the current default device and directory. For other fields that you do not specify, the APPEND command uses the corresponding field of the input file specification.

If you use the asterisk wild card character in any fields of the output file specification, the APPEND command uses the corresponding field of the input file specification. If you are appending more than one input file, APPEND uses the corresponding fields from the first input file.

Command Qualifiers

- /EXCLUDE=(file-spec[,...])

Any files that match the listed file specifications are excluded from the APPEND operation. If you specify only one file, you can omit the parentheses. The file specification can contain a directory specification, but you cannot include the device in the file specifications you supply with the /EXCLUDE qualifier.

Wild card characters are supported for file specifications. However, you cannot use relative version numbers to exclude a specific version.

Examples

1. **$APPEND TEST3.DAT TESTALL.DAT**<CR>

The APPEND command appends the contents of the file TEST3.DAT from the default disk and directory to the file TESTALL.DAT, also located on the default disk and directory.

5.1.2.2 VAX Programs

This section covers all of the programs developed for the Nutrition Division that are of use in data processing. Note that it does not include those files that are an *integral part of the analysis process*, but those that are useful in the day to day manipulation of data. Many of these programs are simply shortcuts, allowing you to perform difficult operations quicker than usual. The following is a brief overview of each of the programs:

5.1.2.2.1	The See Program - Allows you to view the first twenty lines of a file, placing a number line every four lines.
5.1.2.2.2	The Rearrange Program - Allows you to move columns in a file.
5.1.2.2.3	The Fixform Program - Converts a file from variable length records to fixed length ones, or visa versa.

There are three other programs that should be noted here, although their operation is complex enough that they each require their own user manual.

The first is the program DOCFILES. This program allows you to interactively document the contents of a directory. It has many features which simplify and standardize the documentation of a directory, and it stores the information in a file that can be either printed out or scanned for a specific file by using the program. To run this program type **DOCFILES**<CR>.

The next program is the CODMAKER program. Briefly, this program interactively creates, updates, and displays the COD file, along with developing the group codes that are associated with each food item. Aspects of this program will be discussed more in depth in Section 5.1.A-2.

The last program is the Universal Data Entry (UDE) program. This program allows you to define the variable names and data types (see Section 5.1.A-1 for information on variables and data types) of the fields that you wish to have in a data file. It is for entering all types of biochemical information except food consumption data. Of these three programs this is the only one with a completed user manual.

5.1.2.2.1 The See Program

This programs is very useful in that it will help you determine the format of a file. It can handle 80 or 132 column files, automatically switching into the proper mode and displaying the appropriate number line. There are many programs that require you to specify the format of the data file. By typing **SEE <filename>**<CR> you will have the first twenty line displayed, as in the example below:

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
    101   51         0.00      0                 10
    101   55         1.00      9                  0
    101   62         1.00      6                  0
    101   64         1.00      7                  0

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
    101   65         2.00      9                  0
    101   67         0.25      5                  0
    101   68         1.00      9                  0
    101   70         0.00      0                 12

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
    101   73         0.00      0                 14
    101   74         1.00      9                  0
    101   77         1.00      7                  0
    101   80         0.00      0                 12

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
    101   81         0.00      0                 12
    101   82         0.00      0                 12
    101   83         0.00      0                 12
    101   84         0.00      0                 12

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
    101   85         0.00      0                 12
    101   86         0.00      0                 12
    101   87         0.00      0                 12
    101   88         0.00      0                 12
```

In the above example, there is a variable in columns 5-7, one in columns 11-12, one in columns 22-25, one in column 32, and one in columns 50-51. Note that, although the variable in columns 22-25 has only one digit before the decimal point, these are only the first twenty lines and there might be larger values further down in the file. To be safe, you might want to pad the variable and claim that it is in columns 20-25, thus allowing for larger numbers. Consult the person responsible for the format of the specific data file you are working on to determine which fields you should pad, and by how much.

5.1.2.2.2 The Rearrange Program

This program becomes very useful when you wish to rearrange columns in a file. Using the example in Section 5.1.2.2.1, you will note that there are variables in columns 5-7, columns 11-12, columns 20-25, column 32, and columns 50-51. Lets say that you wish to switch the order of the third and fourth variables and that you wish to condense the file so that there are only two spaces between each variable. The first thing that you would do would be to run the SEE program to get the current column positions. The next thing to do would be to translate this information into a Fortran format statement (see Section 5.1.A-1.2). Note that, although the variables you are rearranging are of different types, this requires the variables to be defined as CHARACTER (A) types. Assuming that the original data file is named INFO1.DAT and the resultant data file is to be named INFO2.DAT, a sample of this program would appear as follows:

```
$REARRANGE<CR>

    WHAT IS THE INFILE NAME:  INFO1.DAT<CR>

    HOW MANY VARIABLES ARE TO BE REARRANGED:  5<CR>

    WHAT IS THE INPUT FORMAT: (T5,A3,T11,A2,T20,A6,T32,A1,T50,A2)<CR>

    WHAT IS THE OUTFILE NAME:  INFO2.DAT<CR>

    WHAT INPUT VARIABLE NUMBER IS OUTPUT VARIABLE  1:  1<CR>

    WHAT INPUT VARIABLE NUMBER IS OUTPUT VARIABLE  2:  2<CR>

    WHAT INPUT VARIABLE NUMBER IS OUTPUT VARIABLE  3:  4<CR>

    WHAT INPUT VARIABLE NUMBER IS OUTPUT VARIABLE  4:  3<CR>

    WHAT INPUT VARIABLE NUMBER IS OUTPUT VARIABLE  5:  5<CR>

    WHAT IS THE OUTPUT FORMAT:  (T2,A3,2X,A2,2X,A1,2X,A6,2X,A2)<CR>
```

If you were to type **SEE INFO2.DAT**<CR>, you would get an output similar to the one below:

```
          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 101  51  0    0.00  10
 101  55  9    1.00   0
 101  62  6    1.00   0
 101  64  7    1.00   0

          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 101  65  9    2.00   0
 101  67  5    0.25   0
 101  68  9    1.00   0
 101  70  0    0.00  12

          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 101  73  0    0.00  14
 101  74  9    1.00   0
 101  77  7    1.00   0
 101  80  0    0.00  12

          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 101  81  0    0.00  12
 101  82  0    0.00  12
 101  83  0    0.00  12
 101  84  0    0.00  12

          1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
 101  85  0    0.00  12
 101  86  0    0.00  12
 101  87  0    0.00  12
 101  88  0    0.00  12
```

5.1.2.2.3 The Fixform Program

This program performs the very simple function of converting a file from variable length records to fixed length records, and vice versa. A file of variable length records is one in which each line can be of any length, while a file of fixed length records is one in which all lines must be of the same length. This is a necessary distinction because there are some aspects of Fortran file handling that require a file to have fixed length records, but it is much easier to edit a file of variable length records. Thus, you could edit a file and then convert it to fixed length records to run it through some programs

Be aware that some programs require files of fixed length records. Thus, if a program crashed with an error message to the effect of a file with an inconsistent file extension, the problem might be due to a variable length record file. A sample run of the program follows:

```
$FIXFORM<CR>

  What is the variable length file:  JUNK.DAT<CR>

  What is the fixed length file:  JUNKFIX.DAT<CR>

  Which way are we going today:
  variable to fixed = "v"
  fixed to variable = "f":  V<CR>
```

Note that if an "F" were entered for the last option, the file of fixed length records would have been converted to variable length records.

5.1.3 File Transfer from the PC to the VAX

Note: This process must be performed on a PC that is connected to the VAX via Smarterm 240.

1. The first step is to determine exactly which files must be transferred. This is not always a simple task and is dependant upon the process by which the data were collected. If, for example, the UDE program was used, there are several associated files containing variable names and formats that must be transferred along with the data file. The basic requirement for a file to be transferred is that it contain information that does not exist somewhere else. Thus report files do not need to be transferred, while a COD file does.

2. The next step is to move all of the files to be transferred to one place. However, this requires a hard disk with a substantial amount of free space. If this is not possible then transfer directly from the disk containing the data. Otherwise, create a directory on the hard disk and load into it all of the files to be transferred. To create a directory on the PC, either type **MKDIR** followed by the directory name or use the Norton Utilities feature by typing **NCD**, moving the cursor to the appropriate place, and pressing the **F7** button.

3. At the DOS prompt type **ST240**<CR>. This will connect you to the VAX, where you should log on to the 780. If nothing appears immediately, press the return ke followed by the space bar to wake up the port selector. If this does not w[illegible], try slowly pressing the BREAK key three times followed by the space bar. If you still cannot get the port selector see the local computer expert. Once logged on, move to the directory in which you want the data to be stored. This directory will usually be the data directory of the appropriate study.

4. Once in the correct directory, type **KERMIT**<CR>. This will enable the

Kermit utility, giving you the Kermit prompt (KERMIT-32>).

5. Now type **RECEIVE**<CR>. The cursor will return and simply hang there. At this point press **ALT-Y**. You are now in the Kermit Command Entry utility on the PC.

6. Type the word **SEND** followed by the drive specification, directory path, and file name. If all of the files in the directory are to be sent, the wild card character "*" can be used. Some examples are below:

```
SEND B:\SUBJECT1.DAT<CR>
SEND C:\DATA\*.*<CR>
```

   The terminal will beep when the transfer is completed. To return to Kermit press **ALT-Y**. If you wish to transfer other files, repeat steps 5 and 6. Once you are done transferring files, type **QUIT**<CR> at the Kermit prompt. This will return you to the VAX.

7. Once you are sure that all of the required files have been transferred completely, you can delete the data files from the PC and the disks. From now on the data should be handled exclusively on the VAX. This is not true when the programs to edit and correct the data are specific to the PC. In this case, insure that the data is secure on disks and make backups. Note: data should not be stored permanently on the PC.

5.1.4 Cleanup, Merging, and Analysis of Data

5.1.4.1 General Information

Despite the most rigorous validation, consumption data will never come back entirely clean from the field. The errors may be due to several sources: the data enterer may have mistyped information, the data collector may have made a mistake in coding the data, or the COD file may have inaccurate or wrong information such as the wrong food name or gram weight. These errors need to be identified and corrected regardless of the source, yet keep in mind that a good part of locating errors is identifying their cause. This may entail the use of several associated packages and programs which are discussed in Section 5.1.5.4.3. You do not necessarily have to be familiar with all of these aspects, but be aware of whom to go to for help if problems arise.

5.1.4.2 General Setup of Data

In general, data are organized during studies in such a way that each file contains a logical division of information. For example, each file may contain the consumption information of all subjects that a given data collector observed on a given day. Assuming four data collectors and seven days, there would be twenty-eight files created. In order to connect this consumption data to the nutrient data, all of these files must be merged together. This is a requirement of the Analyze program, which performs its operations in one fell swoop. The merging can be performed simply by the VAX COPY or APPEND commands (sections 5.1.2.1.8 and 5.1.2.1.9, respectively), yet if any global changes or updates need to be performed, a program may be useful (see section 5.1.4.3.1). Regardless of the method, the merged file will contain all of the information that the swarm of smaller files has. Whether these smaller files should be kept permanently or not depends upon the needs of the Primary Investigator. During the cleaning phase of the data processing, these files need to exist as they offer the only way to easily and quickly correct errors in the raw data.

5.1.4.2.1 COD File Setup

The raw data files consist of food intake information and a COD file. The COD file contains the list of all foods used during a study. This does not mean that all foods in the COD file were necessarily eaten, but that they were menu items and snack items available to the subjects. An explanation of each field within the file follows:

Three Digit Code	Indicates the position of the food item in the list of foods. For example, an item with the three-digit code of 95 would be on the 95th line of the list. This approach to organization of the COD file may change in the near future.
Six-character Code	The code by which the nutrient database refers to the food item. For the new system, this code has a maximum width of ten characters. These codes must be correct before the consumption data can be merged with the nutrient information, yet are not necessary for the collection of consumption data.
Food Name	The descriptive name of the item, indicating as precisely as possible the food's identity.
Unit of Measure	Indicates the unit by which the item is served (cups, ounces, serving, etc.) Note that it is not necessarily the serving size per person. If this is the case however, it makes later tasks easier.
Gram Weight of the Unit	This is simply the number of grams that the above unit of measure weighs.

Group Code	This optional field indicates the food group or category the item belongs to. It helps to organize the items in the COD file for easier identification. The specifics of this field are beyond the scope of this document. Consult the Recipe Specialist for in-depth information.
Entry Date	This field contains the latest date that information about the item was changed. It is used to separate out the records that were created or changed within a specific time period.

The data collection and entry program would use the three-digit code to access the other information in the file. For technical details on the COD file, see Appendix G, Section 2.

5.1.4.3 Useful Programs and Their Results

There are several stages in which to isolate errors, yet the general approach is fairly standard throughout analysis; to look at the data in such a way that anomalies stand out. This approach is used from the first stages of merging the data into one cohesive body to the final stage of statistical analysis.

Although many of the features and programs on the VAX, which are useful in data processing, have already been described in section 5.1.2, the following are in-depth discussions of programs specifically developed for error detection and data processing. It tells what is required to take the data from the raw data files to the final stage in which it is ready to be statistically analyzed. File names and general formats listed here are those recommended for future use, however, it may be necessary to modify these names if the study is complex in its organization. Although these programs may reside somewhere else, they should be called from the study's data directory. Failure to do this will result in misplaced files or errors in execution. For information concerning the setup and alteration of the actual programs used, see Appendix G.

5.1.4.3.1 CHANGE Program

This program is used to merge the raw data into one master file when there are global changes and insertions that need to be made before analysis. The exact functions of this program vary from study to study depending upon the requirements of each. One of its main features is to access the COD file and update the fields in the raw data files <u>and</u> the master file based upon the latest version of the COD file. There may be other tasks such as inserting fields or calculating new values. These depend entirely upon the organization of each study. This CHANGE program can also identify those food items that were actually consumed and their frequency of consumption. This helps the recipe specialist in his or her task. The program does not so much point out errors as update and expand the data. It is a useful tool, especially when global changes and updates are desired. It should be noted that if any changes are made to the individual raw data files, the master file needs to be recreated, either through the VAX COPY or APPEND commands, or this program.

The CHANGE program basically merges the raw data files into one master file, performing global changes and updates as it goes. It also may rearrange the order of the variables to one conducive to analysis. The general organization of the program is fairly stable, although specific tasks will vary from study to study. For further information on this program see Appendix G, Section 3. To execute this program type at the prompt, **RUN [<directory path>]CHANGE**. There should not, in general, be anything else required but to wait until it is finished executing. In general, the program uses as input the following files:

> <u><study name>.COD:</u> This file, aspects of which are described in detail in Appendix G, Section 2, contains all the codes for the foods that are used during the study. This is a sequential file that can be typed to the screen or edited to be examined if so desired. For all future studies, the COD file will be managed by an interactive program.
>
> <u>NAMES.LIS:</u> This file contains the names of the raw data files, each on a separate line. It is a sequential file. All the names listed in this file should be considered as primary data files too, for they will be read in during execution.

The following output files are created:

> <u>FOODS.TOT:</u> Contains all of the consumption data for the entire study. It can by typed and edited. It will be used later as input for the ANALYZE program.
>
> <u>RATING.DAT:</u> This optional file is similar to FOODS.TOT except that it lacks all consumption information. It is for use by other divisions or organizations that are interested in the rating and acceptance information captured by the study. It can be typed and edited.

CHANGE must be re-run if any changes were made to the data in order for the output files to reflect these changes when the analysis is done. CHANGE takes about 5 to 15 minutes to run, and you can not continue on to the next step of the analysis unti' [illegible] is complete.

5.1.4.3.2 ANALYZE Program

The diagnostic capability of this program is fairly straightforward. In a nutshell, this program reads in a record from the master food consumption file, searches the Oracle table file created for this specific study for the nutrient information of the food item, calculates the nutrient values for the specific weight of the item, and writes the results to a new file. Running this program will automatically identify those food items that were consumed but not placed in the database. These items should be brought to the attention of the database manager who should correct the database. Note that any corrections in the database require that the Oracle table be re-dumped or the changes will not be reflected in the analysis. Also note that, at present, this program halts execution after it encounters fifty errors. If this limit is reached, be prepared to find more errors in the data. Also note that if this limit is encountered, you have an overabundance of errors and should reexamine your data to ensure it is the correct set of values.

All studies prior to 1989 have a dedicated version of this program in their program directory. If you are re-running any of these old studies, look for a command file of the same name as the ANALYZE program in the program directory. For example, the Ft. Hood study (T8804) has in its program directory the files T8904_ANALYZE.EXE and T8904_ANALYZE.COM. The actual program that performs the analysis is the EXE file, yet the command file (the COM one) will allow you to run the program so that your terminal is not locked. If you do not find a command file, then you will have to run the program itself (**RUN [<directory path>]<program name>**). To run the command file, first make sure that you are in the study's data directory; this is where the necessary data files are located. Then type **@[<directory path>]<command file name>** at the prompt to call the command file. Note that the directory path given should be the study's program directory and not its data directory and that the command file name is the name as *discussed in above.*

For all studies during and after 1989, the ANALYZE program has been somewhat standardized. For these studies you access central ANALYZE programs located in the [NUTRITION.TOOLS.PROGRAMS] directory. Due to the fact that some of the studies conducted in 1989 used information from past studies, several of them use the U Mass Oracle table format. If the ORACLE table organization of the study is of the old, U Mass format, then you type **OLDANAL** at the prompt. If the study uses the new in-house format, type **NEWANAL** at the prompt.

Once you have executed the command file you will be prompted to select how you wish to run the program:

(I)mage, (P)rocess, (R)eprocess

Type "R", for reprocess. DO NOT TYPE "P" or "I", or the program will not be able to read the input file correctly. (Note: "P" is used when you want to enter the file format of the food file for a new study. If you are analyzing the data for the first time, you need to get the format of the data files. See Section 5.1.5.2.2.1 for a program which can do this for you.) In studies previous to 1989 you will be prompted to enter an output filename. A default name, ANALYZE.OUT, should be entered. See the output files for more information on this.

The ANALYZE program uses as input the following files:

FOODS.TOT: This file contains all of the study information grouped together as a single field, the six-character food-code and a quantity indicating the gram weight of the food item consumed. Once again, this can be typed or edited.

<study number>.TAB: A file created by either the program GETTABLE or the program DUMPTABLE, which contains the nutrient information of the foods consumed during the study. It is a keyed index file and cannot be typed or edited. Any time the database is updated, this file must be recreated for the changes to be reflected in analysis. Note that this file must exist before analysis can be performed. See section 5.1.4.3.1 for more information on this file.

The output files created are:

FOODS.OUT: This file contains the descriptive information and the nutrient values associated with each food item from FOODS.TOT. It cannot be typed or edited. This is the final product, ready to be statistically analyzed.

<output file>.OUT: The file contains information on how the program ran. For all studies prior to 1989 the name of the file should be either ANALYZE.OUT or TEMP.OUT. For all studies during and after 1989, the log file name will be either OLDANAL.OUT or NEWANAL.OUT, depending on whether the Oracle table is in the U Mass or the in-house format. This file will serve as the log file to which errors will be written if they occur during execution. It can be typed or edited.

The ANALYZE program will display a screen that essentially is a graphical picture of your process executing. You may wait for the process to end, or you may press CTRL-Y to stop the display and use the terminal to do other work. You MAY NOT perform statistics on the data until this process is complete, which may take from 5 to 15 minutes, depending upon the amount of data in the study.

5.1.4.3.3 The Nutrient Table File

This file is created by either the GETTABLE or the DUMPTABLE program, depending on whetr :r the database is in the old format or the new, respectively. To execute either c these files, move to the study's data directory and type its name at the prompı.

If you are running GETTABLE, you will be asked to supply the name of the table that you wish to create from ORACLE. This is simply the study number (i.e. "T8901") for current studies. Past studies may have the study name as the table name (i.e. "FTSAM"). You will then be asked to supply the name of the file in which to place the information. Simply enter the study number (or name) followed by ".TAB" and you are done.

DUMPTABLE runs similarly, except that you must first choose to dump a nutrient file (option 1 on the menu) before entering the table name.

If an error message occurs after you type the appropriate program name (GETTABLE or DUMPTABLE), you need to re-compile the program. To compile GETTABLE do the following:

```
$CD [NUTRITION.ANALYZE]<CR>
$@GETTABLE<CR>
```

To compile DUMPTABLE type:

```
$CD [NUTRITION.NEWSYS.ROCHE]<CR>
$FOR DUMPTABLE<CR>
$@COM$:LFOR DUMPTABLE<CR>
```

Be aware that this compiles an older version, although quite usable, of the DUMPTABLE program. There is a more user-friendly version in [FINN.TABLES] that is under development and will be completed fairly soon.

5.1.4.3.4 Statistical Analysis

Here begins the actual and final cleaning of the data. A command file, discussed in section 5.1.5.5.3, reads the information in, selects those records that have values outside acceptable ranges, and generates a list of all these records. You must then trace the data back to the raw data. Either confirm or correct the information, recreate the master file, and re-analyze the data. If records are correct and reasonable, yet still have unacceptable values, examine the Oracle database for possible errors in the nutrient values for the associated food item.

5.1.5 Standard Statistical Procedures

5.1.5.1 General Information

When you have cleaned the data of errors, you can then run statistical analyses on the data. Statistics are performed using the SPSSx and BMDP packages. Each study should have a statistics directory and all statistical processes should be contained therein. Any data that exists in other directories and needs to be accessed for statistical analysis should be called from the statistics directory, processed, and any new results should be stored there.

In the initial stages of analysis the data are stored in enormous files. These files are so large that SPSSx can often take several minutes to perform operations upon them. If there are operations that are required by several of the tasks (i.e. grouping the data by subject and day), there is a tendency to set up these operations so that they need to be run only once, storing the results in a new, more compact system file. Then several operations can access this one file and cut out time from their own processing. The drawback to this approach is that the initial files often need to be rerun due to corrections made on the data. If a person unfamiliar with the specific organization of your data is not aware of all of the files that need to be rerun, they may simply run the program using the old data. To avoid this problem, try to keep the number of intermediate files (those that perform simple crunching functions) to a minimum.

5.1.5.2 Information on SPSSx

SPSSx is a comprehensive tool for managing, analyzing and displaying information. Among the features available with SPSSx are:

- Ability to read and analyze complex file and record structures such as indexed, nested, or hierarchical files
- Extensive file management capabilities including the ability to split files, save files, and to match and merge multiple files
- Extended data management capabilities including the ability to define and manipulate long string variables, do conditional data transformations, and perform a wide variety of numeric functions
- Wide range of statistical analysis routines
- Great latitude in data display, incorporating Report as a standard feature

5.1.5.2.1 Running_SPSSx

A variety of qualifiers and parameters are available for inclusion on the SPSSx invocation line. All qualifiers are optional and can be truncated to four or fewer characters according to the VAX/VMS syntax rules for entering DCL commands. Each qualifier must be preceded by a slash (/) and separated from its value by an equals sign (=). The name of the SPSSx command file must be preceded by at least one space. The following examples show several ways of invoking SPSSx:

```
$SPSSx YEAR88.SPS<CR>
$SPSSx/OUTPUT=RESULTS88.LIS YEAR88<CR>
```

```
$SPSSx/OUTPUT=RESULTS88 YEAR88<CR>
$SPSSx/OUTPUT YEAR88<CR>
```

All of the commands above result in SPSSx executing the commands in the file YEAR88.SPS. The first example will send all output to the SYS$OUTPUT device, which is generally the screen. Note that the second example takes advantage of the default file extension for SPSSx command files (.SPS) and specifies the name of the output file. The third example takes advantage of the default file extensions for listing files (.LIS) as well as for command files (.SPS). Finally, the fourth example takes advantage of the default file filename and file extension for OUTPUT (YEAR88.LIS) and the default file extension for SPSSx command files (.SPS). Thus, in brief, the first command puts the results to the screen, the second and third commands place the output in RESULTS88.LIS, and the fourth command places the output in YEAR88.LIS.

5.1.5.2.2 Default_File_Extensions

The following table shows some of the default file extensions for files used by SPSSx.

DEFAULT	Classes or Types of Files
.DAT	DATA LIST Input, KEYED DATA LIST Input, MATRIX DATA Input, Track Input, WRITE Output Files, and procedure output
.JNL	Journal Files
.LIS	Output Files from INFO, PRINT, REPORT, TABLES PRINTER, TABLES 9700, and SPSSx listing files.
.SPS	SPSSx INCLUDE Files and Command Files
.SPSSXSAV	SPSSx System Files, matrix output

5.1.5.3 Stages of Analysis

The following sections describe the appropriate steps required to perform initial analysis on the data. The step must be performed in order since the output from one procedure is the input for the next one.

- FOODSYS.SPS - this SPSSx program will take the analyzed consumption data and put it into a SPSSx system file. It uses as input the file FOODS.OUT (the output of the ANALYZE program). It creates two files as output:

FOODSYS.OUT - a log file of the execution of the program. Check this file to see if any errors occurred. It can be typed or edited.

STUDYFOOD.SYS - this is a SPSSx file that can not be read. It contains all of the information of FOODS.OUT, but you do not have to worry about formats. Variables are referenced simply by name.

To run this program you would type:

$SPSSX/OUT=FOODSYS.OUT FOODSYS.SPS<CR>

- SUBDAY.SPS - SUBDAY.SPS uses the output from FOODSYS and aggregates the information into records of data by subject number and day. This is an example of an intermediate file as described in Section 5.1.5.1. It uses as input the file STUDYFOOD.SYS, the output file from FOODSYS. Note that this program is <u>not</u> necessary, but convenient to use. It simply takes processing time out of later programs. It could be completely skipped in order to streamline the processing of the data. It creates two files as output:

  SUBDAY.OUT - the log file of the execution of the program. Check this file to see if any errors occurred. It can be typed or edited.

  SUBDAY.SPSSXSAV - contains the aggregated data and is used as input for other SPS programs such as ONEWAY.SPS. Can not be typed or edited.

To run this program you would type:

$SPSSX/OUT=SUBDAY.OUT SUBDAY.SPS<CR>

- OUTLIERS.SPS - This file examines the intake values, either per food item or per day, for each subject and reports on those values that are outside of acceptable ranges. It can be setup to use as input either the file STUDYFOOD.SYS or the file SUBDAY.OUT, depending upon how you wish to check you data. It only creates one file as output, OUTLIERS.OUT, which contains the list of all those records with excessively small or large values.

To run this program you would type:

$SPSSX/OUT=OUTLIERS.OUT OUTLIERS.SPS<CR>

There are many other command files that are used to analyzed data. For a more extensive understanding of this process see Chapter 6.0. Also be aware of the fact that this process is constantly under development and is by no means static.

5.1.6 Documentation of a Study's Directories

5.1.6.1 General Information

There are some basic guidelines that should be followed in documenting the directories on the VAX.

- Every program and command file should be internally documented, stating its purpose, creator, date of inception, and explaining in-depth any novel or complex features that it may contain.

- For any processes that involve more than a single basic stage, there should be a file discussing the different aspects of the process, including all associated files and their basic purpose.

- For each set of data files having their own organization, there should be a file discussing the format and other information relating to the data files.

Please be aware that this has not done in the past, although attempts have been made.

5.1.6.2 Docfiles Program

This program, once it is cleaned up and a user manual is developed, will be a very important part of documenting a study. It allows specific file extensions to be documented and automatically stores in the documentation the date the file was documented, the date the file was created and the owner of the file.

This section will be expanded when the program and documentation are completed.

5.2 Analyze Consumption Data User's Guide

5.2.1 Introduction

The Analyze Consumption Data is an interactive program on the VAX for processing food consumption data that has been previously arranged or collected. This document describes how to use the Analyze Consumption Data program. Currently, this document should be considered a "living document".

Food consumption by individuals is recorded in the field and entered into computer data files using the Data Entry Programs in Chapter 4.0. Recipes are created using the Recipe Code/Editor/Analyzer program in existence. The food consumption data is then combined with the nutrient data and analyzed with statistical software. The Analyze Consumption Data program is used to update all data files in which corrections have been made, and to correlate food items and their consumption with their nutrient equivalents.

5.2.2 Getting Started

To run the Analyze Consumption Data program several files are needed. These files must be contained in the same subdirectory. The following is a listing of all the files that are required to be resident in this subdirectory:

Program files:	File Name Description:
ANACONSUM.EXE	The main executable program
ANALYZE.EXE	The analyze program executable
CHANGE.EXE	The change program executable
DOANALYZE.COM	Command file that invokes program
ANALYZE1.COM	Command file to run analyze detached
CHANGE1.COM	Command file to run change detached
ACMENU1.RSC	Main menu template
ANAMENU1.RSC	Analyze menu template
ANAMENU2.RSC	Analyze Process menu template
CHAMENU1.RSC	Change menu template
CHAMENU2.RSC	Change Defaults menu template
ANAMENU1.HLP	Main menu help
ANAMENU2.HLP	Analyze menu help
ANAMENU3.HLP	Analyze Process menu help
CHAMENU1.HLP	Change menu help
CHAMENU2.HLP	Change Defaults menu help
FUNCTIONS.FH	Include file of menu functions
MENUS.FH	Include file of menu structures
RECVARS.FH	Include file of variables

To start the program, type at the DCL prompt:

```
$ ANACONSUM
```

This is a VAX symbol which will invoke the Analyze Consumption Data program. You may be in any directory when you run the program, but it is important to be cognizant of file privilege limitations.

5.2.3 The Main Menu

The main menu currently contains 3 items. To make selections use the cursor keys or the RETURN key to move the selection bar up and down. Press ENTER on the numeric keypad to make the selection. Note that the selection bar wraps around from top to bottom and vice versa.

On-line help exists for all menus. Press /H ENTER to view it, or if you are using a VT-220 compatible terminal or higher, you may also use the DO key in place of ENTER and the HELP key instead of /H ENTER.

The first item of the main menu is:

"CHANGE consumption data"

Select this option to initialize the update of all data files due to changes in the collected data.

The second item of the main menu is:

"ANALYZE the consumption data"

Select this to combine the consumption data with the recipe or food item nutrient data.

The third item of the main menu is:

"QUIT program"

Select this option if you want to exit the Analyze Consumption Data program. A prompt will display asking you to verify that you want to quit by pressing PF2. Any other key board entry at this time will bring you back to the main menu.

5.2.3.1 Change Consumption Data

When "CHANGE" the consumption data is selected from the main menu, the Choose Change Options menu will display. This menu has two options:

"Use the current default file names"
"Change/view the default names"

Default values for input and output files of the CHANGE program are located in a file with the name CHANGE.FRM. Upon execution of the CHANGE program, the file names that contain input data are read in and also a field that tells where to write the output data. You may press the PF2 key to return to the main menu at this time. Since the CHANGE program must process so much data, its execution time can exceed 10 to 15 minutes. If you remain in the Analyze Consumption Data program, you will be notified when the process has completed. At this time you will be informed to select HELP in order to view the LOG file. This LOG file is created during the execution of the CHANGE program, and it contains information about the process, such as when it started, ended, or if any errors occurred. It is suggested that you view this file now, because you will not be able to see this file again after this time.

FIGURE 5.2 Use the current default file names

Selecting this option will cause the immediate execution of the CHANGE program as a detached process. The defaults contained in CHANGE.FRM will be used.

FIGURE 5.3 Change/view the default file names

Selecting this option will cause the Change Defaults Menu to display. At this time, the defaults file CHANGE.FRM will be read in, and the existing input and file names will be displayed with their appropriate field names. You may change as many of the fields that you desire. Move from field to field with arrow keys. When you are done changing the defaults, press ENTER. This will cause the immediate execution of the CHANGE program as a detached process, as well as causing the old defaults file to be overwritten. If you did not make any changes, or decide for whatever the reason not to use your changes, press PF2 to bring you back to the main menu.

5.2.3.2 Analyze the Consumption Data

Selecting Analyze at the Main Menu will cause the Analyze Menu to appear. This menu has two options:

"PROCESS the consumption data"
"REPROCESS the consumption data"

Default values for input and output files, and for input format and missing value are located in file called ANA.FRM. Upon execution of the ANALYZE program, the file is used as input to obtain these default values. You may press PF2 at this time if you do not wish to continue. Since the ANALYZE program must process so much data, its execution time can exceed 10 to 15 minutes. If you remain in the Analyze Consumption Data program, you will be notified when the process has completed. At this time you will be informed to select HELP in order to view the LOG file. This LOG file is created during the execution of the ANALYZE program, and it contains information about the process, such as when it started, ended, or if any errors occurred. It is suggested that you choose to view this file now because you will not be able to see this file after this time.

FIGURE 5.4 Process the consumption data

Select this option if you wish to process the consumption data. The Process Menu will display, and the default values for the fields on the screen will be obtained from the ANA.FRM file. You may change any, all, or none of the fields. Use the arrow keys to move from field to field. When you are finished entering your new default values, press ENTER. This will cause the immediate execution of the ANALYZE program as a detached process, and the defaults file will be overwritten. If, for any reason, you decide not to use your changes or have made none, press PF2 to return to the main menu.

FIGURE 5.5 Reprocess the consumption data

Select this option if you wish to reprocess the consumption data using the current default values contained in the file ANA.FRM. This will cause the immediate execution of the ANALYZE program as a detached process.

5.3 EXAMPLE: Alaska Consumption Data Analysis Procedures

This section is a detailed example of the steps involved in analyzing a ration study. The documentation includes all of the processes involved in entering, changing, analyzing, and running statistics on the Alaska study data. All input and output files will be noted where necessary, along with their format. It is recommended that the user be logged on the VAX in the NUTRITION account before performing these procedures, since many of the procedures create files so large, that the disk quota will be exceeded and the process will terminate before completion.

5.3.1 Dealing with Corrected Raw Data

It is very likely that errors or bad entries occurred when the data entry program was being used. In either case, the data must be corrected. These changes are accomplished by using the same study specific data entry program (or the universal DE program in the case of ALASKA) to edit the raw data files.

Once the data has been corrected, programs must be executed to merge output data files. The program called CHANGE will accomplish the initial step of this process for the user. To perform CHANGE, do the following:

1) At the "$" prompt, type ALAS2DAT

This is a symbol that will move you to the [NUTRITION.T8901.DATA] directory (NOTE: to see a translation of a VAX symbol, type SHOW SYMBOL then the name of the symbol at the prompt).

2) type CHANGE at the prompt. This is another symbol that will execute the CHANGE program.

5.3.2 Description of the CHANGE Program

The CHANGE program uses as input the following files:

1) [NUTRITION.T8901.DATA]ALASKA.COD
   -this is a COD file that contains all the codes for the foods (MREs) that were used during the Alaska study. This is a sequential file that can be typed to the screen or edited to be examined if so desired.

2) [NUTRITION.T8901.DATA]NAMES.LIS
   -contains on separate lines the names of the daily Alaska data files to be used as data input. These files contain consumption data. It is a sequential file. All the names listed in this file should be considered as primary data files too, for they will be read in during execution.

The output files created are:

1) [NUTRITION.T8901.DATA]FOODS.TOT
   -all the foods and their consumption quantity are now linked together in this file, which can be typed or edited. It will be used later as input for the ANALYZE program.

2) [NUTRITION.T8901.DATA]RATING.DAT
   -contains the hedonic ratings of individual food items. It can be typed and edited.

CHANGE must be re-run if any changes were made to the data in order for these to be incorporated into the analysis. CHANGE takes about 5 to 15 minutes to finish, and you can not continue on to the next step of the analysis until it is finished. You can, however, work on other projects. The "$" prompt will appear immediately, even though the process is still actually executing. If you type SHOW SYSTEM at the prompt, you will be able to see if the program is still processing - it has the process name CHANGE. When this no longer appears when you SHOW

SYSTEM, the process is completed.

5.3.3 Analyze the Consumption Data

The combining of the consumption data with the nutrient data is executed from the same directory as the CHANGE program is, so retype the symbol ALAS2DAT at the prompt.

Next type the symbol ALAS2ANAL. You have now invoked the command file to execute the program. You will be prompted to select how you wish to run the program:

(I)mage, (P)rocess, (R)eprocess

Type "R", for reprocess. DO NOT TYPE "P" or "I", for this will cause the program to not be able to read the input file correctly (Note: "P" is used when you want to enter the file format of the food file for a new study).

You will then be prompted to enter an output filename. This file will serve as the log file where errors will be written during execution of the program. The names of DOANALYZE.OUT or ANALYZE.OUT have been used previously, and should be chosen to be consistent.

The ANALYZE *program calculates nutrients for the specific amounts of food* consumed contained as listed in the input file.

Input files:

1) [NUTRITION.T8901.DATA]FOODS.TOT
   -file containing a six character food code and a quantity representing the gram weight of the food item consumed. Once again, this can be typed or edited.

2) [NUTRITION.T8901.DATA]T8901.TAB
   -table created to be compatible with ORACLE. Can not be typed or edited. Contains six digit food code and quantity consumed.

Output files:

1) [NUTRITION.T8901.DATA]FOODS.OUT
   -file of the analyzed consumption data. Can not be typed or edited. Contains six digit food code, quantity consumed, and nutrient values.

2) [NUTRITION.T8901.DATA]DOANALYZE.OUT
   -contains list of errors encountered in the program. Can be typed or edited to see how process ran.

The ANALYZE program will display a screen that is a graphical picture of your process executing. You may wait for the process to end, or you may press CTRL-Y to stop the display to work on other projects. You MAY NOT perform STATS until this process is complete, which can take from 5 to 15 minutes.

5.3.4 Running Statistical Analysis on the Data

When you are sure that all the data is fine and the previous procedures have been properly executed, you can then run statistical analyses on the data. The statistics of the Alaska data were created using the SPSSx package. To run statistics type ALAS2STAT at the prompt. This will move you to the [NUTRITION.T8901.STAT] directory, where all the SPSSx programs reside.

If changes have been made, it will be necessary to execute two of the statistics programs, and they are:

1) FOODSYS.SPS - this SPSSx program will put the analyzed consumption data into SPSSX format.

It uses as input:
[NUTRITION.T8901.DATA]FOODS.OUT - see previous section.

and creates as output:
[NUTRITION.T8901.STAT]FOODSYS.OUT
-log file of the execution of the program. Check here for errors in data or program. It can be typed or edited.

[NUTRITION.T8901.STAT]STUDYFOOD.SYS
-this is a SPSSx file that can not be read.

To run FOODSYS type at the prompt:

```
$ SPSSX/OUT=FOODSYS.OUT FOODSYS.SPS
```

2) SUBDAY.SPS - SUBDAY uses the output from FOODSYS (FOODSYS.OUT) and aggregates all the records of data by subject number and day.

it uses as input:
[NUTRITION.T8901.STAT]STUDYFOOD.SYS
-the output file from FOODSYS.SPS

creates as output:
[NUTRITION.T8901.STAT]SUBDAY.OUT
-the log file of the execution of the program. Check here for

errors. It can be typed or edited.

[NUTRITION.T8901.STAT]SUBDAY.SPSSXSAV
-contains the aggregated data and is used as input for other statistical programs such as ONEWAY.SPS. Can not be typed or edited.

To run SUBDAY type at the prompt:
```
$ SPSSX/OUT=SUBDAY.OUT SUBDAY.SPS
```

5.3.5 Description of Alaska Data Files

The following sections describes the formats of the files that are connected with the Alaska study. Many of these files are used in a similar form in connection with other nutrition studies, and so these descriptions may be useful outside of the scope of the Alaska study.

5.3.5.1 The COD File

In each study, a COD (short for CODE) file is created prior to field data collection. This file contains a list of (hopefully) all the foods that will be encountered during the study in the format described below:

FILE: ALASKA.COD

<u>FOOD NUMBER</u> <u>FOODCODE</u> <u>CODE FIELD</u> <u>FOODNAME</u> <u>GRAMWEIGHT</u>

FOOD NUMBER	-	Each food in the file has a sequential number starting with 1 for the first line, and ending with the number of actual food items. Usually three digits in length.
FOOD CODE	-	Unique Code identifying a specific food. It is an alphanumeric field (ex. MRE120) that matches a coded food item with a nutrient database item.
CODE FIELD	-	Optional field to categorize foods into groups.
FOODNAME	-	The character description of the food item, such as MASHED POTATOES.
GRAMWEIGHT	-	Gram weight per serving of the food item.

5.3.5.2 The Field Data Files

The data collected in the field is usually broken up into different files according to GROUP and DATE. Then, all the names of these files are collected in a file, so that during analysis, each of these files can be opened and used as input data. The format of these individual data files is described below:

FILE: GR4DAY02FD.DAT

SUBJECT NUMBER   FOODNUM   AMT. CONSUMED   RATING   REASON NOT EATEN

SUBJECT NUMBER	-	A number that is unique to a subject in the study.
FOODNUM	-	The food code for the food that was consumed, corresponding to the FOOD NUMBER in the COD file.
AMT. CONSUMED	-	Portion that was eaten (real number)
RATING	-	A numeric value, ex. 1-9, with 9 indicating the most acceptable food.
REASON NOT EATEN	-	A numeric code that represents a reason that a subject did not eat or finish a food item.

5.3.5.3 The NAMES.LIS File

As stated previously, all this ASCII sequential file contains is a line by line list of the data files.

5.3.5.4 The Merged Data File: FOODS.TOT

The execution of the program CHANGE (see document: Alaska Consumption Data Analysis Procedures) will create this data file. This file correlates the food item with the six digit food codes that are contained in the COD file. The final format of this file is as follows:

FILE: FOODS.TOT

GRP DATE SUBNUM RAT REASON NE FNUM FNAME FDCODE GRAMWT AMTCNS

GRP	-	Group that the subject was in. This number is obtained from the input file name.
DATE	-	The day, in two digits, also obtained from the input file name.

SUBNUM	-	Short for subject number. Unique number of the subject.
RAT	-	Short for rating, the rating of the food item by the subject.
REASON NE	-	Short for reason not eaten, number that represents a reason why a food item was not eaten or totally consumed.
FNUM	-	The food number based on COD file number of a food item that corresponds to six digit food code.
FDNAME	-	The character description of the food item. Obtained from COD file.
FDCODE	-	The alphanumeric code of the food, obtained from the COD file, whose nutrient data is needed.
GRAMWT	-	The weight of a serving of food in grams, obtained from COD file.
AMTCNS	-	Amount of food item consumed by the subject.

5.3.5.5 The RATING.DAT File

The rating file is in actuality a subset of the FOODS.TOT file. It is created, like FOODS.TOT, upon the execution of the CHANGE program. The file format is described below:

FILE: RATING.DAT

GRP	DATE	SUBNUM	RAT	REASON NE	FNUM	FDNAME

See the descriptions for these fields in the FOODS.TOT file description.

CHAPTER 6.0

JOB CONTROL LANGUAGE SAMPLES FOR STATISTICAL ANALYSES

Contributing Authors:
Dr. Donald Poe
Dr. Michael Sutherland

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                       Page   1
13:51:15    USARIEM                         on VAX780::           VMS V4.7

VAX-11/780              USARIEM                              License Number 17731
This software is functional through September 30, 1989.

Try the new SPSS-X Release 3.0 and 3.1 features:

* Interactive SPSS-X command execution          * The new RANK procedure
* Online,VMS-like Help                          * Improvements in:
* Nonlinear Regression                          *   REPORT and TABLES
* Time Series and Forecasting (TRENDS)          *   Simplified Syntax
* Macro Facility                                *   Matrix I/O

See SPSS-X User's Guide, Third Edition, for more information on these features.

   1  0  FILE HANDLE SYSIN/NAME='[NUTRITION.T8901.STAT]SUBDAY.SPSSXSAV'
   2  0
   3  0  COMMENT  ALL SPSSX INSTRUCTIONS FILES MUST BEGIN WITH A FILE HANDLE
   4  0           LINE.  THE ABOVE LINE USES THE SYSIN CONVENTION TO TELL THE
   5  0           COMPUTER THAT YOU WILL BE CALLING IN A SYSTEMS FILE.  THIS
   6  0           IS BECAUSE THE DATA THAT NUTRITION STUDIES TYPICALLY USE
   7  0           ARE CREATED BY OTHER FILES, SUCH AS FOOD INTAKE FILES, AND
   8  0           ARE NOT A PART OF THE INSTRUCTION, OR COMMAND, FILES THEMSELVES.
   9  0           THE DATA THAT THIS ANALYSIS WILL BE USING ARE CREATED BY A
  10  0           FILE CALLED SUBDAY.SPSSXSAV, AND THIS PARTICULAR FILE RESIDES
  11  0           IN THE STAT SUBDIRECTORY OF A NUTRITION DIRECTORY CALLED T8901,
  12  0           OR THE FIRST STUDY OF 1989, OR THE 1989 ALASKA STUDY.
  13  0
  14  0           NOTE ALSO THAT YOU CAN AT ANY TIME INSERT COMMENT LINES INTO A
  15  0           FILE BY WRITING THE WORD COMMENT ON THE LEFT HAND MARGIN, AND
  16  0           TYPING IN YOUR NOTES.  THE ONLY REQUIREMENT HERE IS THAT THE
  17  0           COMMENTS THEMSELVES BE INDENTED FROM THE LEFT MARGIN AT LEAST
  18  0           ONE SPACE.  FURTHER NOTE THAT SPSSX WILL READ ALL CAPS OR LOWER
  19  0           CASE TYPE.
  20  0
  21  0  GET FILE  SYSIN
  22  0      /MAP
  23  0
  24  0
  25  0  COMMENT WHILE THE FILE HANDLE LINE SIMPLY TELLS THE COMPUTER WHAT DATA
  26  0          FILE YOU WILL BE USING, THE ABOVE TWO LINES CAUSE THE COMPUTER
  27  0          TO ACTUALLY GO AND GET THE DATA FILE REFERRED TO ON THE FILE
  28  0          HANDLE LINE.  FURTHERMORE THE MAP LINE WILL PRODUCE A DISPLAY ON
  29  0          THE OUTPUT THAT SHOWS WHAT VARIABLES FROM THE DATA FILE THAT
  30  0          HAVE BEEN PULLED OUT FOR ANALYSIS.  IN THIS CASE WE DID NOT USE THE
  31  0          KEEP SUBCOMMAND TO PULL OUT ONLY SELECTED VARIABLES, AND SO ALL OF
  32  0          THE VARIABLES WILL BE RETAINED FOR POSSIBLE USE.  THE FOLLOWING TABLE
  33  0          SHOWS THE RESULTS OF THIS MAPPING.  NOTE THAT WE COULD ALSO HAVE
  34  0          USED THE RENAME SUBCOMMAND TO RENAME VARIABLES, BUT IN THIS CASE
  35  0          THIS WAS NOT DONE, AND THE RESULT IS THAT ALL VARIABLES ARE PULLED
  36  0          IN FOR ANALYSIS, AND ALL VARIABLES RETAIN THEIR ORIGINAL NAMES.
  37  0
  38  0

File DISK$USER2:[NUTRITION.T8901.STAT]SUBDAY.SPSSXSAV;
   Label:  AGGREGATED FILE
   Created:  26-APR-89 10:15:44 - 32 variables

FILE MAP

Result    Input1     Result    Input1     Result    Input1     Result    Input1
------    ------     ------    ------     ------    ------     ------    ------
GROUP     GROUP      H2O       H2O        VITC      VITC       ETOT      ETOT
SUBNUM    SUBNUM     CAL       CAL        THIAMIN   THIAMIN    PHOSPHOR  PHOSPHOR
DATE      DATE       PROT      PROT       RIB       RIB        MAGNES    MAGNES
RATING    RATING     FAT       FAT        NIACIN    NIACIN     IRON      IRON
RSNNE     RSNNE      CARBO     CARBO      VITB6     VITB6      ZINC      ZINC
PORTWT    PORTWT     CALC      CALC       FOLACIN   FOLACIN    NA        NA
AMTCON    AMTCON     CRF       CRF        VITB12    VITB12     CLORIDE   CLORIDE
QUANTITY  QUANTITY   ASH       ASH        VITAIU    VITAIU     K         K
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                                  Page   2
13:51:18    USARIEM                         on VAX780::           VMS V4.7

  39  0  SORT CASES BY GROUP
  40  0
  41  0
  42  0
  43  0
  44  0  COMMENT  IN THE ALASKA STUDY, THE SUBJECTS WERE RUN IN GROUPS.  IN THIS CASE
  45  0          THERE WERE FOUR GROUPS AND THE ABOVE LINE INSTRUCTS THE COMPUTER TO
  46  0          ORDER THE SUBJECTS BY THEIR GROUP NUMBER.  IN THIS CASE, SINCE WE
  47  0          HAVE NOT TOLD IT TO DO OTHERWISE, THE COMPUTER HAS SORTED THE CASES
  48  0          BY GROUPS IN ASCENDING ORDER.  THAT IS, THE GROUP GIVEN THE LOWEST
  49  0          NUMBER CODE COMES FIRST, FOLLOWED BY THE REMAINING THREE GROUPS IN
  50  0          ASCENDING ORDER.  THIS IS THE DEFAULT IN THE SORT COMMAND, BUT IT
  51  0          CAN BE OVERRIDDEN SO AS TO BE IN DESCENDING ORDER.  SEE INSTRUCTIONS
  52  0          IN THE MANUAL FOR THE SORT COMMAND IF YOU WISH TO DO THIS.
  53  0
  54  0          NOTE ALSO THAT WE COULD HAVE SORTED ON TWO OR MORE VARIABLES RATHER
  55  0          THAN ON JUST THE ONE.  TO DO THIS SIMPLY ADD MORE VARIABLE NAMES TO
  56  0          THE SORT LIST.  FOR EXAMPLE, IF WE HAD SAID:
  57  0
  58  0              SORT CASES BY GROUP DATE MEAL SUBNUM
  59  0
  60  0          THE COMPUTER WOULD HAVE SORTED THE CASES BY GROUP FIRST (IN ASCENDING
  61  0          ORDER), THEN BY DATE STARTING WITH THE EARLIEST, THEN BY MEAL CODE,
  62  0          AND FINALLY BY SUBJECT NUMBER.  THIS IS A SINGLE SORT, NOT MULTIPLE.
  63  0          THIS RESULTS, FOR EXAMPLE, IN A FILE CONTAINING FIRST
  64  0
  65  0              GROUP 1, DAY 1, MEAL 1, SUBJECT 1        FOLLOWED BY
  66  0              GROUP 1, DAY 1, MEAL 1, SUBJECT 2        ETC. THROUGH
  67  0                                                       ALL SUBJECTS.  THEN COMES
  68  0              GROUP 1, DAY 1, MEAL 2, SUBJECT 1        ETC.
  69  0
  70  0          NOTE ALSO THAT ALTHOUGH THE WORD "SORT" IS AN SPSSX COMMAND, IT IS ONLY
  71  0          READ AS A COMMAND BY SPSSX WHEN IT APPEARS ON THE LEFT HAND MARGIN,
  72  0          AND HENCE IN THIS CASE, IT IS WITHIN A COMMENT AND NOT ON THE MARGIN,
  73  0          THE COMPUTER DOES NOT EXECUTE A SECOND SORT.  THIS GENERAL RULE IS TRUE
  74  0          OF ALL SPSSX COMMANDS.
  75  0
SIZE OF FILE TO BE SORTED:      1164 CASES OF     256 BYTES EACH.
SORT COMPLETED SUCCESSFULLY.  FILE SIZE:        583 BLOCKS.

Preceding task required 9.70 seconds CPU time;  23.03 seconds elapsed.
  76  0  DESCRIPTIVES VARIABLES = H2O TO FAT/ STATISTICS = ALL
  77  0
  78  0  COMMENT THE ABOVE LINE USES THE DESCRIPTIVES COMMAND TO PRODUCE DESCRIPTIVE
  79  0          STATISTICS ON ALL VARIABLES.  THE RESULTING TABLE REPRESENTS VALUES
  80  0          OF DESCRIPTIVE STATISTICS FOR ALL SUBJECTS, ALL MEALS, ALL DAYS, AND
  81  0          ALL GROUPS COMBINED.  TO GET DESCRIPTIVE STATISTICS ON SELECTED SUBSETS
  82  0          OF GROUPINGS, USE THE TEMPORARY COMMAND, FOLLOWED BY THE SELECT IF
  83  0          COMMAND. AN EXAMPLE FOLLOWS IN WHICH WE GET DESCRIPTIVE STATISTICS FOR
  84  0          GROUP 1, DAY 4.
  85  0
  86  0          SINCE WE HAVE ASKED FOR ALL AVAILABLE STATISTICS, THE COMPUTER
  87  0          CANNOT PRINT THIS INFORMATION ON ONE LINE PER VARIABLE (NEW
  88  0          STYLE PRINTING), AND MUST USE MULTIPLE LINES PER VARIABLE.  HENCE
  89  0          WE WILL GET A WARNING BELOW TELLING US OF THIS FACT.  THIS IS
  90  0          EXPECTED AND IS NOT A WORRY.
  91  0

>Warning # 11003
>The new default column-style printing cannot be used for this DESCRIPTIVES, as
>there are too many statistics to print on one line per variable.  Old style
>printing will be used instead.

There are 467,808 bytes of memory available.

296 bytes of memory required for the DESCRIPTIVES procedure.
8 bytes have already been acquired.
288 bytes remain to be acquired.
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                        Page   3
13:51:50    USARIEM                     on VAX780::           VMS V4.7

File:       AGGREGATED FILE


Number of valid observations (listwise) =       1164.00

Variable  H2O

Mean            474.549             S.E. Mean        6.682                Std Dev        227.979
Variance      51974.601             Kurtosis          .907                S.E. Kurt         .143
Skewness           .336             S.E. Skew         .072                Range         1739.093
Minimum            .00              Maximum        1739.09                Sum         552374.621

Valid observations -      1164       Missing observations -       0

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Variable  CAL

Mean           2837.788             S.E. Mean       37.526                Std Dev       1280.286
Variance    1639132.308             Kurtosis         -.200                S.E. Kurt         .143
Skewness           .240             S.E. Skew         .072                Range         8611.895
Minimum            .00              Maximum        8611.90                Sum        3303184.686

Valid observations -      1164       Missing observations -       0

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Variable  PROT

Mean            114.745             S.E. Mean        1.397                Std Dev         47.661
Variance       2271.542             Kurtosis          .403                S.E. Kurt         .143
Skewness           .084             S.E. Skew         .072                Range          346.956
Minimum            .00              Maximum         346.96                Sum         133563.287

Valid observations -      1164       Missing observations -       0

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Variable  FAT

Mean            119.625             S.E. Mean        1.578                Std Dev         53.851
Variance       2899.971             Kurtosis          .088                S.E. Kurt         .143
Skewness           .301             S.E. Skew         .072                Range          354.938
Minimum            .00              Maximum         354.94                Sum         139243.382

Valid observations -      1164       Missing observations -       0

Preceding task required 4.21 seconds CPU time;  8.84 seconds elapsed.
```

```
  92  0  TEMPORARY
  93  0  SELECT IF (GROUP EQ 1 AND DATE EQ 4)
  94  0  DESCRIPTIVES VARIABLES = H2O TO FAT/ STATISTICS = ALL
  95  0

>Warning # 11003
>The new default column-style printing cannot be used for this DESCRIPTIVES, as
>there are too many statistics to print on one line per variable.  Old style
>printing will be used instead.


There are 467,552 bytes of memory available.


296 bytes of memory required for the DESCRIPTIVES procedure.
8 bytes have already been acquired.
288 bytes remain to be acquired.
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                      Page   5
13:51:56    USARIEM                        on VAX780::          VMS V4.7

File:     AGGREGATED FILE

Number of valid observations (listwise) =        32.00

Variable  H2O

Mean           299.213             S.E. Mean     37.209             Std Dev       210.485
Variance     44303.781             Kurtosis       -.931             S.E. Kurt        .809
Skewness          .504             S.E. Skew       .414             Range         703.800
Minimum           3.08             Maximum       706.88             Sum          9574.816

Valid observations -       32        Missing observations -       0

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Variable  CAL

Mean          1740.162             S.E. Mean    190.025             Std Dev      1074.946
Variance   1155508.351             Kurtosis       -.192             S.E. Kurt        .809
Skewness          .807             S.E. Skew       .414             Range        3754.327
Minimum         297.77             Maximum      4052.10             Sum         55685.187

Valid observations -       32        Missing observations -       0

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Variable  PROT

Mean            74.726             S.E. Mean      8.373             Std Dev        47.364
Variance      2243.378             Kurtosis      -1.201             S.E. Kurt        .809
Skewness          .358             S.E. Skew       .414             Range         156.342
Minimum           8.26             Maximum       164.61             Sum          2391.240

Valid observations -       32        Missing observations -       0

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Variable  FAT

Mean            80.042             S.E. Mean      8.910             Std Dev        50.402
Variance      2540.381             Kurtosis       -.026             S.E. Kurt        .809
Skewness          .860             S.E. Skew       .414             Range         176.691
Minimum          16.29             Maximum       192.98             Sum          2561.358

Valid observations -       32        Missing observations -       0

Preceding task required 3.31 seconds CPU time;  5.49 seconds elapsed.
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                              Page   6
13:51:56    USARIEM                        on VAX780::          VMS V4.7

  96  0  MEANS TABLES H2O TO FAT BY GROUP
  97  0
  98  0  COMMENT THE ABOVE LINE INSTRUCTS THE COMPUTER TO PRINT OUT THE MEANS OF ALL
  99  0          VARIABLES IN THE FILE FROM H2O TO POTASSIUM FOR EACH OF THE FOUR GROUPS.
 100  0          THE FOLLOWING TABLES ARE THE RESULT OF THIS COMMAND.  AS OUTPUT WE GET
 101  0          ONE MEANS TABLE FOR EACH OF THE NUTRIENTS AND THE INFORMATION PRINTED
 102  0          INCLUDES THE MEAN, STANDARD DEVIATION AND NUMBER OF CASES ON WHICH
 103  0          THESE CALCULATIONS ARE BASED.  SINCE IT PRINTS OUT INFORMATION BROKEN
 104  0          DOWN BY ONLY ONE VARIABLE, THIS IS KNOWN AS A ONE-WAY MEANS TABLE.
 105  0          OPTIONS AVAILABLE WITH THE MEANS COMMAND INCLUDE VARIOUS WAYS TO DEAL
 106  0          WITH MISSING DATA (THE DEFAULT IS TO DELETE CASES WITH MISSING DATA ON
 107  0          A TABLE-WIDE BASIS), AND YOU CAN ALSO GET ADDITIONAL STATISTICS BY
 108  0          ASKING FOR THEM HERE.  SEE THE MANUAL FOR DETAILS UNDER THE MEANS
 109  0          PROCEDURE.
 110  0
 111  0
 112  0

There are 467,936 bytes of memory available.

Given workspace allows for 9,613 cells of 1 dimensions for BREAKDOWN
```

```
File:     AGGREGATED FILE

      D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable    H2O
    Broken Down by    GROUP       Ration Group


Variable       Value  Label                      Mean     Std Dev     Cases

For Entire Population                         474.5486   227.9794      1164

GROUP              1                          341.7393   205.3265       282
GROUP              2                          559.0777   248.7278       239
GROUP              3                          414.0853   190.0705       301
GROUP              4                          578.2015   184.2894       342

  Total Cases = 1164

File:     AGGREGATED FILE

      D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable    CAL
    Broken Down by    GROUP       Ration Group


Variable       Value  Label                      Mean     Std Dev     Cases

For Entire Population                        2837.7875  1280.2860      1164

GROUP              1                         2009.1928  1176.9192       282
GROUP              2                         2801.5586  1254.1355       239
GROUP              3                         2830.2842  1078.1466       301
GROUP              4                         3552.9365  1116.6196       342

  Total Cases = 1164
```

```
File:     AGGREGATED FILE

       D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable    PROT
    Broken Down by    GROUP      Ration Group


Variable       Value  Label                         Mean      Std Dev     Cases

For Entire Population                             114.7451    47.6607      1164

GROUP              1                               86.5776    47.7601       282
GROUP              2                              113.7368    49.0305       239
GROUP              3                              118.1451    43.1272       301
GROUP              4                              135.6832    38.0346       342

  Total Cases = 1164

File:     AGGREGATED FILE

       D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable    FAT
    Broken Down by    GROUP      Ration Group


Variable       Value  Label                         Mean      Std Dev     Cases

For Entire Population                             119.6249    53.8514      1164

GROUP              1                               93.1687    56.2892       282
GROUP              2                              118.2799    54.6684       239
GROUP              3                              122.6457    47.2797       301
GROUP              4                              139.7209    47.2636       342

  Total Cases = 1164
```

```
Preceding task required 5.05 seconds CPU time;  7.64 seconds elapsed.

 113  0  MEANS  TABLES = H2O TO PROT BY GROUP BY SUBNUM
 114  0
 115  0  COMMENT THE ABOVE LINE INSTRUCTS THE COMPUTER TO GENERATE A TWO-WAY MEANS
 116  0          TABLE.  THAT IS, THE DATA ARE SUMMARIZED BY SUBJECT WITHIN EACH
 117  0          GROUP.  NOTE THAT TO SAVE SPACE WE HAVE ASKED THAT THIS BE DONE FOR
 118  0          ONLY THREE NUTRIENTS (THOSE FROM H2O TO PROTEIN IN THE MAPPING LIST,
 119  0          OR H2O, CALORIES AND PROTEIN).  THE FOLLOWING THREE TABLES PRESENT THE
 120  0          RESULT OF THIS INSTRUCTION.
 121  0
 122  0
 123  0

There are 467,936 bytes of memory available.

Given workspace allows for 7,865 cells of 2 dimensions for BREAKDOWN
```

```
26-Sep-89    SPSS-X RELEASE 3.1 FOR VAX/VMS                                                      Page  10
13:52:26     USARIEM                          on VAX780::           VMS V4.7

File:      AGGREGATED FILE

        D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable    H2O
    Broken Down by    GROUP      Ration Group
                by    SUBNUM     Subject Number

Variable          Value  Label                       Mean     Std Dev      Cases

For Entire Population                            474.5486    227.9794       1164

GROUP                 1                          341.7393    205.3265        282
  SUBNUM            101                          376.0137    205.4476          8
  SUBNUM            102                          341.2724    157.7863          9
  SUBNUM            103                          228.4900    204.1193          9
  SUBNUM            104                          294.6192    163.5414         10
  SUBNUM            105                          405.4553    398.6977          9
  SUBNUM            106                          284.0699    112.9946          9
  SUBNUM            107                          174.4311    116.6204          9
  SUBNUM            108                          330.8894    206.1052          4
  SUBNUM            109                          242.0842    205.3845          7
  SUBNUM            110                          545.4400    222.4567         10
  SUBNUM            111                          555.1779    326.7793          9
  SUBNUM            112                          216.1313    113.7128          9
  SUBNUM            113                          537.9692    135.3007         10
  SUBNUM            114                          319.5056     79.0792          9
  SUBNUM            115                          220.8883    118.0684         10
  SUBNUM            116                          299.4523    121.1399          7
  SUBNUM            117                          457.3947    160.7612          8
  SUBNUM            118                          483.1766    155.7267         10
  SUBNUM            119                          396.8198    163.7083          8
  SUBNUM            120                          455.7601    166.4323         10
  SUBNUM            122                          449.0068    165.2993         10
  SUBNUM            123                          273.0759    210.0648         10
  SUBNUM            124                          167.6853     85.2972          7
  SUBNUM            125                          462.7340    151.6557          6
  SUBNUM            126                          205.9789    177.0856          9
  SUBNUM            127                          205.1969    105.7693         10
  SUBNUM            128                          233.6500    138.5190          8
  SUBNUM            129                          228.9877    137.0812          9
  SUBNUM            130                          331.3188    139.5179          8
  SUBNUM            131                          299.8654    208.1157          8
  SUBNUM            132                          344.7343    117.2028          6
  SUBNUM            134                          442.9685    192.9508          9
  SUBNUM            135                          414.3434    232.3307          7
  SUBNUM            136                          303.1138       .0000          1

GROUP                 2                          559.0777    248.7278        239
  SUBNUM            201                          713.4317    309.8180          3
  SUBNUM            203                          561.6874    145.0810         10
  SUBNUM            204                          679.0591    149.8687          5
  SUBNUM            205                         1111.7445    887.2048          2
  SUBNUM            206                          378.3815    242.1703          7
  SUBNUM            207                          588.1572    187.7946         10
  SUBNUM            208                          453.5673    239.3538          4
  SUBNUM            209                          499.1712     59.1944          5
  SUBNUM            210                          564.8190    115.7802         10
  SUBNUM            211                          414.0858    196.5553          9
  SUBNUM            213                          549.9258    191.2664          5
  SUBNUM            214                          629.8912    112.4861         10
  SUBNUM            215                          931.1502    215.6594          8
  SUBNUM            216                          539.5611    179.5075          6
  SUBNUM            218                          547.8275    165.3415          5
  SUBNUM            219                          198.6041    161.5191          7
  SUBNUM            220                          348.4452    182.8364          9
  SUBNUM            221                          279.3928     82.4166          7
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                           Page  11
13:52:27    USARIEM                        on VAX780::            VMS V4.7

Criterion Variable H2O

Variable          Value  Label                        Mean      Std Dev      Cases

  SUBNUM            222                           489.3464     117.0244          3
  SUBNUM            223                           831.4779     407.3478          7
  SUBNUM            224                           524.2125     119.3901          4
  SUBNUM            225                           751.2306      46.7464          8
  SUBNUM            226                           562.1512     247.3470          9
  SUBNUM            227                           268.4249     182.5562          4
  SUBNUM            228                           474.2832     210.5909          6
  SUBNUM            229                           685.6293     231.8377          6
  SUBNUM            230                           682.9422     143.5599         10
  SUBNUM            231                           563.8614     266.1990          6
  SUBNUM            232                           538.9604     203.9329          6
  SUBNUM            233                           796.6799      21.1514          2
  SUBNUM            234                           506.5738     272.6010          9
  SUBNUM            235                           574.3060     172.1091          6
  SUBNUM            236                           729.6812     101.8186          7
  SUBNUM            237                           388.6908     162.5256          7
  SUBNUM            238                           392.2385     172.5303          4
  SUBNUM            239                           680.5695     100.6586          8
  SUBNUM            240                           539.5234     283.4511          5

GROUP                 3                           414.0853     190.0705        301
  SUBNUM            301                           592.3965     174.1545         10
  SUBNUM            303                           500.8168     212.0030         10
  SUBNUM            304                           545.2006     198.1464          2
  SUBNUM            305                           344.3828     124.9306         10
  SUBNUM            306                           459.3475      98.7950         10
  SUBNUM            307                           419.4308     194.5175         10
  SUBNUM            308                           513.6212     131.5203         10
  SUBNUM            309                           167.7708     124.0368         10
  SUBNUM            310                           341.2965     151.5028          2
  SUBNUM            311                           520.4326     115.3967         10
  SUBNUM            312                           353.2483      85.8675         10
  SUBNUM            313                           407.8079     134.4884          9
  SUBNUM            314                           512.6053     269.9996         10
  SUBNUM            315                           356.2238     184.0133          8
  SUBNUM            316                           433.1448     172.3810         10
  SUBNUM            317                           600.3744     170.2153         10
  SUBNUM            318                           311.9548     149.6457         10
  SUBNUM            319                           285.7868     160.1754          4
  SUBNUM            320                            95.3150     110.9586          2
  SUBNUM            321                           550.9898     286.1371          8
  SUBNUM            322                           307.5020     200.3223          9
  SUBNUM            323                           260.5615     144.7784          8
  SUBNUM            324                           390.2856     185.0542         10
  SUBNUM            325                           367.1904     137.4164         10
  SUBNUM            326                           381.1847      89.0150         10
  SUBNUM            327                           252.8090     100.8958         10
  SUBNUM            328                           301.3246     134.1336          9
  SUBNUM            329                           580.4491     103.8104         10
  SUBNUM            330                           449.1858     213.3258         10
  SUBNUM            331                           439.0995     130.4843         10
  SUBNUM            332                           401.4960     179.3167         10
  SUBNUM            333                           526.2948     217.2637         10
  SUBNUM            334                           454.8266     173.5220         10
  SUBNUM            336                           331.7502     128.5690         10

GROUP                 4                           578.2015     184.2894        342
  SUBNUM            401                           821.5643     163.2034         10
  SUBNUM            402                           474.4698     146.2036         10
  SUBNUM            403                           481.1524     125.2373         10
  SUBNUM            404                           581.3927     161.9026         10
  SUBNUM            405                           664.1901      96.5340         10
  SUBNUM            406                           411.4942     246.4859          7
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                        Page  12
13:52:27    USARIEM                        on VAX780::          VMS V4.7

Criterion Variable H2O

Variable          Value  Label                          Mean     Std Dev      Cases

   SUBNUM          407                              564.2917    183.8893         10
   SUBNUM          408                              714.1386    113.5393         10
   SUBNUM          409                              335.7471    112.7917          7
   SUBNUM          410                              380.8491     80.9773          2
   SUBNUM          411                              565.8367    203.2217         10
   SUBNUM          412                              647.6478    116.5227          9
   SUBNUM          413                              524.3914     83.6000          9
   SUBNUM          414                              567.3639    170.5540         10
   SUBNUM          415                              477.8221    134.7480         10
   SUBNUM          416                              526.2589    232.8168          2
   SUBNUM          417                              637.5758     65.6660         10
   SUBNUM          418                              472.1937    158.6233          9
   SUBNUM          419                              371.1063    202.7330         10
   SUBNUM          420                              591.3610    126.1434          9
   SUBNUM          421                              737.2707    211.2568          9
   SUBNUM          422                              503.9466    300.3659         10
   SUBNUM          423                              779.8063    207.9569         10
   SUBNUM          424                              523.9322    193.1226         10
   SUBNUM          425                              671.1359    100.6930          9
   SUBNUM          426                              474.1393    107.7276          9
   SUBNUM          427                              456.8389    149.7578         10
   SUBNUM          428                              609.8374    194.1519         10
   SUBNUM          429                              494.6559    148.0418         10
   SUBNUM          430                              579.9620    153.8445         10
   SUBNUM          431                              641.8985     59.2639         10
   SUBNUM          432                              668.4491    128.2818         10
   SUBNUM          433                              688.9218     86.1072         10
   SUBNUM          434                              648.4307     65.7112         10
   SUBNUM          435                              612.4683    126.1758         10
   SUBNUM          436                              682.1890    122.9569         10
   SUBNUM          437                              551.3295    117.6084         10
   SUBNUM          439                                1.0268       .0000          1

   Total Cases = 1164
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                             Page  13
13:52:28    USARIEM                        on VAX780::         VMS V4.7

File:     AGGREGATED FILE

        D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable   CAL
    Broken Down by   GROUP      Ration Group
                by   SUBNUM     Subject Number

Variable         Value  Label                       Mean     Std Dev      Cases

For Entire Population                          2837.7875   1280.2860       1164

GROUP                1                         2009.1928   1176.9192        282
  SUBNUM           101                         1617.1982    864.4261          8
  SUBNUM           102                         2110.4900   1006.6040          9
  SUBNUM           103                         1539.6510    915.4966          9
  SUBNUM           104                         1402.2819    643.7717         10
  SUBNUM           105                         2519.1435   1192.9971          9
  SUBNUM           106                         1647.8114    493.7831          9
  SUBNUM           107                         1046.2475    586.2929          9
  SUBNUM           108                         2540.5433   1247.0369          4
  SUBNUM           109                          863.4101    544.7229          7
  SUBNUM           110                         3536.4175   1663.4951         10
  SUBNUM           111                         3810.3184   1486.0220          9
  SUBNUM           112                         1257.6745    513.5087          9
  SUBNUM           113                         2965.7245    568.8540         10
  SUBNUM           114                         1765.7137    409.2099          9
  SUBNUM           115                         1076.5710    551.4488         10
  SUBNUM           116                         2084.2285    923.3238          7
  SUBNUM           117                         2172.8990    667.7935          8
  SUBNUM           118                         2170.6149    544.7524         10
  SUBNUM           119                         2882.4530   1115.2269          8
  SUBNUM           120                         2860.1740   1152.5914         10
  SUBNUM           122                         2462.1985   1167.2765         10
  SUBNUM           123                         1346.2374    774.7376         10
  SUBNUM           124                         1037.4637    383.3133          7
  SUBNUM           125                         2373.7790    258.1765          6
  SUBNUM           126                         1165.3379    653.1126          9
  SUBNUM           127                         3667.6810   1829.0386         10
  SUBNUM           128                         1484.7833   1268.3522          8
  SUBNUM           129                         1161.9410    668.4856          9
  SUBNUM           130                         1849.7028    609.0366          8
  SUBNUM           131                         1676.4695    734.6807          8
  SUBNUM           132                         1876.1636    570.7032          6
  SUBNUM           134                         1912.2366    841.4512          9
  SUBNUM           135                         2019.3870    460.1087          7
  SUBNUM           136                         1122.9725       .0000          1

GROUP                2                         2801.5586   1254.1355        239
  SUBNUM           201                         3735.6040    931.2097          3
  SUBNUM           203                         2605.7991    633.3768         10
  SUBNUM           204                         3478.3502    881.5958          5
  SUBNUM           205                         4374.4113   2943.0113          2
  SUBNUM           206                         1743.4998    909.9323          7
  SUBNUM           207                         2521.3368    723.8006         10
  SUBNUM           208                         2126.3482   1127.1835          4
  SUBNUM           209                         3502.1296    257.4929          5
  SUBNUM           210                         3487.7590    733.4935         10
  SUBNUM           211                         1494.6657    824.8720          9
  SUBNUM           213                         2857.0788   1425.2984          5
  SUBNUM           214                         3453.3930    895.7316         10
  SUBNUM           215                         4569.8463    831.3904          8
  SUBNUM           216                         1508.4970    481.7326          6
  SUBNUM           218                         2799.6694   1149.6490          5
  SUBNUM           219                         1606.0322    854.7464          7
  SUBNUM           220                         2265.4693   1032.7925          9
  SUBNUM           221                         1723.3601    533.0924          7
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                          Page  14
13:52:29    USARIEM                        on VAX780::         VMS V4.7

Criterion Variable CAL

Variable          Value  Label                          Mean      Std Dev      Cases


  SUBNUM           222                             2981.3842   746.6864          3
  SUBNUM           223                             4542.8951  2256.7740          7
  SUBNUM           224                             3656.8822  1139.0279          4
  SUBNUM           225                             3312.6772   550.6290          8
  SUBNUM           226                             2080.3794   793.8809          9
  SUBNUM           227                             1590.8014   851.0601          4
  SUBNUM           228                             2259.4601   759.2985          6
  SUBNUM           229                             3291.1336  1116.1651          6
  SUBNUM           230                             3594.8363   753.1400         10
  SUBNUM           231                             2157.0741   959.4196          6
  SUBNUM           232                             2733.5779  1348.5190          6
  SUBNUM           233                             4283.0011  1110.9420          2
  SUBNUM           234                             2664.2652  1652.3325          9
  SUBNUM           235                             3166.6788   953.7647          6
  SUBNUM           236                             2525.3793   436.1218          7
  SUBNUM           237                             2115.1602   927.6140          7
  SUBNUM           238                             1946.2203   906.1405          4
  SUBNUM           239                             3938.9760   550.7634          8
  SUBNUM           240                             2680.1000  1214.4005          5

GROUP                3                             2830.2842  1078.1466        301
  SUBNUM           301                             4405.6818  1416.2883         10
  SUBNUM           303                             1894.3789   791.8600         10
  SUBNUM           304                             3159.4538   302.4041          2
  SUBNUM           305                             2074.4659   538.0701         10
  SUBNUM           306                             4056.3393   864.1063         10
  SUBNUM           307                             3442.9009  1095.3166         10
  SUBNUM           308                             3764.4287   708.3910         10
  SUBNUM           309                             1820.3644   664.0358         10
  SUBNUM           310                             1988.9120   656.7109          2
  SUBNUM           311                             3721.8898   730.6917         10
  SUBNUM           312                             3123.6725   680.0550         10
  SUBNUM           313                             2094.6275   523.4968          9
  SUBNUM           314                             3733.2286   981.5535         10
  SUBNUM           315                             2809.6969   597.6901          8
  SUBNUM           316                             3220.3142   722.1343         10
  SUBNUM           317                             3705.2593   744.7943         10
  SUBNUM           318                             2962.2159   967.7046         10
  SUBNUM           319                             2035.1569   534.3681          4
  SUBNUM           320                             1399.3139  1152.4117          2
  SUBNUM           321                             2538.4760   863.2414          8
  SUBNUM           322                             2135.3468   822.2339          9
  SUBNUM           323                             1621.7881   539.3812          8
  SUBNUM           324                             2768.9917  1018.0863         10
  SUBNUM           325                             2468.7398   576.0320         10
  SUBNUM           326                             3512.4814   802.7616         10
  SUBNUM           327                             1901.2529   492.1240         10
  SUBNUM           328                             1497.7471   372.4592          9
  SUBNUM           329                             3159.3596   804.1540         10
  SUBNUM           330                             2310.6963   546.6207         10
  SUBNUM           331                             3333.4323   787.6102         10
  SUBNUM           332                             3080.7124   916.8524         10
  SUBNUM           333                             2948.2893   711.6019         10
  SUBNUM           334                             3047.9614  1080.9868         10
  SUBNUM           336                             1879.9792   484.3843         10

GROUP                4                             3552.9365  1116.6196        342
  SUBNUM           401                             4298.5417   841.0707         10
  SUBNUM           402                             2751.2215   499.1143         10
  SUBNUM           403                             2606.3190   486.2479         10
  SUBNUM           404                             3297.6910   720.4163         10
  SUBNUM           405                             3433.6923   470.3534         10
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                                Page  15
13:52:29    USARIEM                       on VAX780::          VMS V4.7

Criterion Variable CAL

Variable          Value  Label                    Mean     Std Dev      Cases

   SUBNUM           406                      1845.6776   1104.9077          7
   SUBNUM           407                      3698.3636   1339.3797         10
   SUBNUM           408                      4199.6257    431.8373         10
   SUBNUM           409                      2277.0855    406.8482          7
   SUBNUM           410                      2452.2361   1316.0603          2
   SUBNUM           411                      3586.5573   1325.9505         10
   SUBNUM           412                      4233.7493    930.2812          9
   SUBNUM           413                      3204.8940    724.2404          9
   SUBNUM           414                      3847.1852    648.2053         10
   SUBNUM           415                      3078.2151    920.2202         10
   SUBNUM           416                      3663.1028    953.1228          2
   SUBNUM           417                      4021.0937    643.9602         10
   SUBNUM           418                      2959.1603    802.0803          9
   SUBNUM           419                      2158.6871   1022.9770         10
   SUBNUM           420                      3158.3314    479.8347          9
   SUBNUM           421                      3243.5568    816.3243          9
   SUBNUM           422                      3169.3553   1561.2305         10
   SUBNUM           423                      5353.8030    878.9827         10
   SUBNUM           424                      3246.8773   1023.2983         10
   SUBNUM           425                      3459.4987    537.8950          9
   SUBNUM           426                      3848.5313    983.6033          9
   SUBNUM           427                      2867.7229   1217.6473         10
   SUBNUM           428                      2928.2523   1186.6168         10
   SUBNUM           429                      3618.9211    974.2781         10
   SUBNUM           430                      3594.9044    493.0523         10
   SUBNUM           431                      4499.0931    187.7614         10
   SUBNUM           432                      4142.6137    792.1212         10
   SUBNUM           433                      5117.8205    596.3708         10
   SUBNUM           434                      4304.6099    592.5604         10
   SUBNUM           435                      3743.4604    945.3705         10
   SUBNUM           436                      4409.1627    810.2188         10
   SUBNUM           437                      3711.7853    440.8134         10
   SUBNUM           439                       189.0091       .0000          1

   Total Cases = 1164
```

```
26-Sep-89    SPSS-X RELEASE 3.1 FOR VAX/VMS                                                    Page  16
13:52:30     USARIEM                        on VAX780::          VMS V4.7

File:     AGGREGATED FILE

        D E S C R I P T I O N   O F   S U B P O P U L A T I O N S

Criterion Variable   PROT
    Broken Down by   GROUP      Ration Group
                by   SUBNUM     Subject Number


Variable        Value  Label                      Mean     Std Dev      Cases

For Entire Population                         114.7451     47.6607       1164

GROUP               1                          86.5776     47.7601        282
  SUBNUM          101                          82.4669     43.5690          8
  SUBNUM          102                          95.4844     43.3752          9
  SUBNUM          103                          51.9343     36.4802          9
  SUBNUM          104                          65.9499     31.0653         10
  SUBNUM          105                         127.4651     85.7424          9
  SUBNUM          106                          70.7932     18.8165          9
  SUBNUM          107                          43.3000     26.9985          9
  SUBNUM          108                          99.2990     47.5764          4
  SUBNUM          109                          57.9410     44.3089          7
  SUBNUM          110                         140.3903     54.6989         10
  SUBNUM          111                         138.1251     66.1094          9
  SUBNUM          112                          53.1773     19.5088          9
  SUBNUM          113                         141.9045     28.5456         10
  SUBNUM          114                          83.0047     19.7360          9
  SUBNUM          115                          50.0311     24.3965         10
  SUBNUM          116                          78.3366     29.1333          7
  SUBNUM          117                         107.6612     45.4694          8
  SUBNUM          118                         105.2556     30.5918         10
  SUBNUM          119                         116.9425     50.9545          8
  SUBNUM          120                         113.7554     43.3964         10
  SUBNUM          122                         116.7897     40.5318         10
  SUBNUM          123                          60.9452     41.0789         10
  SUBNUM          124                          45.1246     14.7595          7
  SUBNUM          125                         108.5766     17.6203          6
  SUBNUM          126                          59.7753     41.9153          9
  SUBNUM          127                          91.3968     36.5013         10
  SUBNUM          128                          57.1271     41.6782          8
  SUBNUM          129                          56.1233     35.1481          9
  SUBNUM          130                          84.2300     34.4260          8
  SUBNUM          131                          77.7207     40.9026          8
  SUBNUM          132                          79.4776     21.9434          6
  SUBNUM          134                          88.4465     35.8639          9
  SUBNUM          135                          98.5548     36.3156          7
  SUBNUM          136                          47.6139       .0000          1

GROUP               2                         113.7368     49.0305        239
  SUBNUM          201                         140.9930     54.6427          3
  SUBNUM          203                         116.0708     26.7949         10
  SUBNUM          204                         138.1099     34.2189          5
  SUBNUM          205                         213.5236    175.8020          2
  SUBNUM          206                          70.1389     36.1936          7
  SUBNUM          207                         103.1145     27.7290         10
  SUBNUM          208                          92.6182     45.2432          4
  SUBNUM          209                         133.8182      7.2259          5
  SUBNUM          210                         138.7208     30.4381         10
  SUBNUM          211                          76.9476     41.0014          9
  SUBNUM          213                         109.8973     50.3161          5
  SUBNUM          214                         138.3889     29.1606         10
  SUBNUM          215                         180.4290     22.3669          8
  SUBNUM          216                          85.7190     29.6351          6
  SUBNUM          218                         111.1006     31.7636          5
  SUBNUM          219                          63.0494     41.1715          7
  SUBNUM          220                          79.7738     39.7751          9
  SUBNUM          221                          65.3032     16.7940          7
```

```
26-Sep-89    SPSS-X RELEASE 3.1 FOR VAX/VMS                                              Page  17
13:52:31     USARIEM                      on VAX780::          VMS V4.7

Criterion Variable PROT

Variable          Value  Label                      Mean     Std Dev      Cases

  SUBNUM            222                         124.9517     25.5042          3
  SUBNUM            223                         181.4706     84.9972          7
  SUBNUM            224                         152.9196     44.8280          4
  SUBNUM            225                         132.1551     16.0630          8
  SUBNUM            226                          90.3443     38.3563          9
  SUBNUM            227                          66.2877     42.6314          4
  SUBNUM            228                          99.7444     32.7410          6
  SUBNUM            229                         132.9773     34.3041          6
  SUBNUM            230                         136.3688     21.8022         10
  SUBNUM            231                          85.2835     45.5189          6
  SUBNUM            232                         103.3918     37.8280          6
  SUBNUM            233                         168.3973     20.2778          2
  SUBNUM            234                         111.4173     68.0629          9
  SUBNUM            235                         126.8339     40.4456          6
  SUBNUM            236                         120.6625     20.5420          7
  SUBNUM            237                          87.0778     36.4107          7
  SUBNUM            238                          71.8821     42.0836          4
  SUBNUM            239                         137.3951     16.7830          8
  SUBNUM            240                         110.3038     60.2566          5

GROUP                 3                         118.1451     43.1272        301
  SUBNUM            301                         168.6283     48.1668         10
  SUBNUM            303                         126.4486     41.4078         10
  SUBNUM            304                         152.4018     32.6789          2
  SUBNUM            305                          89.4133     28.3500         10
  SUBNUM            306                         153.5923     30.9689         10
  SUBNUM            307                         137.6553     52.6192         10
  SUBNUM            308                         145.8060     35.6910         10
  SUBNUM            309                          62.9608     36.6172         10
  SUBNUM            310                          90.8504     48.1350          2
  SUBNUM            311                         152.1645     25.3336         10
  SUBNUM            312                         118.6281     34.9145         10
  SUBNUM            313                         106.5124     30.4089          9
  SUBNUM            314                         141.0010     52.4832         10
  SUBNUM            315                          98.6260     20.9558          8
  SUBNUM            316                         129.8044     37.4160         10
  SUBNUM            317                         147.8620     31.6823         10
  SUBNUM            318                         110.1861     34.1226         10
  SUBNUM            319                          99.1732     32.4042          4
  SUBNUM            320                          60.6420     36.5785          2
  SUBNUM            321                         122.9783     53.2005          8
  SUBNUM            322                          90.1917     48.3787          9
  SUBNUM            323                          75.4133     23.6362          8
  SUBNUM            324                         110.8187     42.8406         10
  SUBNUM            325                         112.1043     27.1287         10
  SUBNUM            326                         128.9319     31.1768         10
  SUBNUM            327                          83.8193     23.2398         10
  SUBNUM            328                          76.7506     26.2869          9
  SUBNUM            329                         141.8092     29.2358         10
  SUBNUM            330                         118.5493     41.0654         10
  SUBNUM            331                         130.7515     31.7456         10
  SUBNUM            332                         123.6064     31.8603         10
  SUBNUM            333                         132.4389     40.6514         10
  SUBNUM            334                         114.2089     44.4356         10
  SUBNUM            336                          90.8083     25.7703         10

GROUP                 4                         135.6832     38.0346        342
  SUBNUM            401                         146.5858     14.8373         10
  SUBNUM            402                         118.8660     21.3661         10
  SUBNUM            403                         103.9437     25.1737         10
  SUBNUM            404                         130.4252     26.9069         10
  SUBNUM            405                         130.9947     17.4893         10
  SUBNUM            406                          88.3921     54.9657          7
```

```
26-Sep-89    SPSS-X RELEASE 3.1 FOR VAX/VMS                                                          Page  18
13:52:32     USARIEM                          on VAX780::          VMS V4.7

Criterion Variable PROT

Variable          Value  Label                          Mean      Std Dev      Cases

  SUBNUM            407                             130.4345     43.3489         10
  SUBNUM            408                             159.9831     19.7110         10
  SUBNUM            409                              88.3183     15.7158          7
  SUBNUM            410                              97.2046     38.8773          2
  SUBNUM            411                             141.3831     44.4649         10
  SUBNUM            412                             154.2725     27.0313          9
  SUBNUM            413                             125.8254     23.5488          9
  SUBNUM            414                             135.9335     25.6654         10
  SUBNUM            415                             116.4434     34.2995         10
  SUBNUM            416                             134.5788     39.3497          2
  SUBNUM            417                             153.2012     14.5590         10
  SUBNUM            418                             108.7961     26.9351          9
  SUBNUM            419                              95.0850     48.1818         10
  SUBNUM            420                             140.5942     22.9164          9
  SUBNUM            421                             132.8113     35.7739          9
  SUBNUM            422                             103.0190     46.6823         10
  SUBNUM            423                             197.1158     32.4831         10
  SUBNUM            424                             120.1825     36.3596         10
  SUBNUM            425                             142.2381     18.4446          9
  SUBNUM            426                             147.0186     28.4998          9
  SUBNUM            427                             111.9728     44.8424         10
  SUBNUM            428                             126.5543     39.2245         10
  SUBNUM            429                             136.0263     29.0741         10
  SUBNUM            430                             136.2327     19.2292         10
  SUBNUM            431                             160.3252      7.1202         10
  SUBNUM            432                             157.8962     16.1951         10
  SUBNUM            433                             189.3159     18.3178         10
  SUBNUM            434                             167.9180     16.2350         10
  SUBNUM            435                             147.3434     31.0361         10
  SUBNUM            436                             160.4787     26.5810         10
  SUBNUM            437                             135.9740     13.2287         10
  SUBNUM            439                               2.7546       .0000          1

  Total Cases = 1164
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                              Page 19
13:52:33    USARIEM                            on VAX780::          VMS V4.7

Preceding task required 16.05 seconds CPU time;  28.98 seconds elapsed.

 124  0  MEANS VARIABLES = GROUP(1,4) DATE(1,3) H2O TO PROT(LO,HI)
 125  0      /CROSSBREAK = H2O TO PROT BY GROUP BY DATE
 126  0
 127  0  COMMENT WHILE THE FIRST TWO SETS OF MEANS OUTPUTS PRESENTED ABOVE
 128  0          ARE RUN IN WHAT IS KNOWN AS GENERAL MODE, THERE IS ALSO
 129  0          ANOTHER MODE, KNOWN AS INTEGER MODE, WHICH YOU CAN USE TO
 130  0          GET SIMILAR OUTPUT.  IN THIS CASE YOU NEED TO PROVIDE THE
 131  0          COMPUTER WITH A LIST OF VARIABLES THAT YOU INTEND TO USE
 132  0          IN PRODUCING MEANS TABLES, ALONG WITH INFORMATION ABOUT THE
 133  0          MINIMUM AND MAXIMUM VALUES OF THE GROUPINGS THAT YOU WANT TO
 134  0          USE.  THIS INFORMATION IS GIVEN INSIDE THE PARENTHESES ABOVE,
 135  0          SO THAT, FOR EXAMPLE, THE MINIMUM VALUE OF THE GROUP VARIABLE
 136  0          IS A 1, WHILE THE MAXIMUM VALUE IS A 4.  IF YOU WANT TO USE
 137  0          ONLY GROUPS 1 THROUGH 3, SIMPLY CHANGE THE 4 TO A 3 IN THE ABOVE
 138  0          LINE.
 139  0
 140  0          INTEGER MODE IS USUALLY FASTER THAN GENERAL MODE, AND CAN BE
 141  0          DONE MORE EFFICIENTLY, ALTHOUGH THERE ARE SOME DIFFERENCES
 142  0          IN OUTPUT AND IN VARIABLE ENTRY ORDER THAT MAY PUSH YOU TO
 143  0          USE GENERAL MODE.  MOST OF THE TIME INTEGER MODE IS PREFERABLE.
 144  0
 145  0          THE ADDITION OF THE CROSSBREAK LINE ADDS A NEW DIMENSION TO
 146  0          TABLE BUILDING.  IF WE HAD LEFT IT OFF, WE WOULD HAVE GOTTEN
 147  0          OUTPUT THAT IS EXACTLY LIKE THE TWO-WAY MEANS TABLE PRODUCED
 148  0          ABOVE BY THE GENERAL MODE COMMAND.  INSTEAD WE GET, AS THE
 149  0          FOLLOWING THREE TABLES SHOW, OUTPUT THAT IS PHYSICALLY DIVIDED
 150  0          INTO CELLS.  EACH OF THE CELLS INCLUDES INFORMATION ON THE MEAN
 151  0          VALUE OF THE VARIABLE (I.E., H2O, CALORIES OR PROTEIN) FOR THAT
 152  0          CELL, THE NUMBER OF OBSERVATIONS ON WHICH THE MEAN IS BASED, AND
 153  0          THE STANDARD DEVIATION.  ADDITIONAL INFORMATION PER CELL IS
 154  0          AVAILABLE BY USING THE CELLS SUBCOMMAND.  OPTIONAL STATISTICS
 155  0          AVAILABLE HERE INCLUDE AN ANOVA AND TESTS OF LINEARITY.  SEE
 156  0          THE MANUAL.  FOR SIMPLICITY HERE WE HAVE USED INFORMATION FOR
 157  0          ONLY THE FIRST THREE DAYS OF THE STUDY, HENCE THE (1,3) AFTER
 158  0          DATE.
 159  0

Integer BREAKDOWN problem requires 1032 bytes of memory.

There are 467,936 bytes of memory available.
```

```
File:      AGGREGATED FILE

                         C R O S S - B R E A K D O W N

Criterion Variable   H2O
    Broken Down by   GROUP        Ration Group
                by   DATE         Date

                    DATE
            Mean |
           Count |                                               Row
         Std Dev |                                              Total
                 |     1    |     2    |     3    |
GROUP   ---------+----------+----------+----------+
               1 | 362.3948 | 421.6903 | 410.6194 | 397.3682
                 |       34 |       32 |       31 |       97
                 | 133.9763 | 176.8218 | 273.9902 | 201.0302
                -+----------+----------+----------+
               2 | 427.0723 | 648.6201 | 606.8196 | 573.4728
                 |       24 |       33 |       31 |       88
                 | 155.8234 | 295.5416 | 183.4611 | 242.0805
                -+----------+----------+----------+
               3 | 407.5005 | 329.3127 | 286.2231 | 343.4680
                 |       33 |       32 |       29 |       94
                 | 140.7693 | 170.6707 | 177.7418 | 169.0716
                -+----------+----------+----------+
               4 | 552.5111 | 530.0917 | 536.5446 | 539.5252
                 |       33 |       35 |       33 |      101
                 | 123.3355 | 228.6855 | 204.9069 | 190.1953
                -+----------+----------+----------+
     Column Total  437.5124   484.7710   464.0891   462.6009
                        124        132        124        380
                   154.4013   251.7704   243.6323   222.0095

784 Missing Observations


Criterion Variable   CAL
    Broken Down by   GROUP        Ration Group
                by   DATE         Date

                    DATE
            Mean |
           Count |                                               Row
         Std Dev |                                              Total
                 |     1    |     2    |     3    |
GROUP   ---------+----------+----------+----------+
               1 |1958.8995 |2457.0781 |2266.3723 |2221.5116
                 |       34 |       32 |       31 |       97
                 | 813.1156 |1073.6075 |1319.1421 |1089.4227
                -+----------+----------+----------+
               2 |2275.9484 |3159.8062 |2986.8667 |2857.8322
                 |       24 |       33 |       31 |       88
                 |1063.8022 |1192.0508 |1096.8745 |1170.7427
                -+----------+----------+----------+
               3 |263[illegible].7184 |2112.7749 |2187.5414 |2320.1298
                 |       33 |       32 |       29 |       94
                 | [illegible]3[illegible].0936 | 921.6604 |1016.5198 | 944.7682
                -+----------+----------+----------+
               4 |3212.4140 |3175.9148 |3184.5860 |3190.6734
                 |       33 |       35 |       33 |      101
                 | 822.3560 |1332.5361 |1371.4209 |1192.5716
                -+----------+----------+----------+
     Column Total  2534.5138  2739.8933  2672.4230  2650.8581
                        124        132        124        380
                   989.3919  1221.7048  1278.5505  1171.0892

784 Missing Observations
```

```
File:     AGGREGATED FILE

                     C R O S S - B R E A K D O W N

Criterion Variable   PROT
    Broken Down by   GROUP      Ration Group
                by   DATE       Date

                    DATE
              Mean |
             Count |                                        Row
           Std Dev |                                        Total
                   |     1    |     2    |     3    |
GROUP     ---------+----------+----------+----------+
                 1 |  87.5192 | 107.2862 | 102.5972 |  98.8590
                   |       34 |       32 |       31 |       97
                   |  28.3624 |  41.2442 |  63.7424 |  46.5754
                  -+----------+----------+----------+
                 2 |  93.0391 | 134.1018 | 122.2657 | 118.7333
                   |       24 |       33 |       31 |       88
                   |  38.2848 |  55.7140 |  37.2872 |  47.7995
                  -+----------+----------+----------+
                 3 | 119.0104 |  96.4435 |  87.3429 | 101.5583
                   |       33 |       32 |       29 |       94
                   |  33.5999 |  39.4728 |  46.1348 |  41.5780
                  -+----------+----------+----------+
                 4 | 134.6988 | 124.7174 | 120.8535 | 126.7162
                   |       33 |       35 |       33 |      101
                   |  26.7092 |  45.9828 |  47.5164 |  41.2729
                  -+----------+----------+----------+
     Column Total    109.5241   115.9835   108.8053   111.5334
                          124        132        124        380
                      36.7458    47.8863    51.0387    45.6823

784 Missing Observations
```

```
Preceding task required 3.70 seconds CPU time;  7.54 seconds elapsed.

 160  0  FINISH

        160 command lines read.
          0 errors detected.
          2 warnings issued.
         45 seconds CPU time.
         87 seconds elapsed time.
      End of job.
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                          Page   1
14:32:23    USARIEM                         on VAX780::          VMS V4.7

VAX-11/780             USARIEM                            License Number 17731
This software is functional through September 30, 1989.

Try the new SPSS-X Release 3.0 and 3.1 features:

* Interactive SPSS-X command execution          * The new RANK procedure
* Online,VMS-like Help                          * Improvements in:
* Nonlinear Regression                          *   REPORT and TABLES
* Time Series and Forecasting (TRENDS)          *   Simplified Syntax
* Macro Facility                                *   Matrix I/O

See SPSS-X User's Guide, Third Edition, for more information on these features.

   1  0  TITLE DAILY MEAN INTAKE USING MISSING DATA.
   2  0
   3  0  FILE HANDLE  INFILE/ NAME = '[NUTRITION.T8805.STAT]SUPER.SYS'
   4  0  FILE HANDLE  SYSFLE/ NAME = '[NUTRITION.T8805.STAT]MISSING.SYS'
   5  0
   6  0  GET FILES = INFILE/KEEP = DATE SUB MEAL CODE LOCAT ENERGY PROTEIN FAT
   7  0                 CHOLESTR CARBOHYD VITA THIAMIN RIBOFLAV
   8  0                 NIACIN VITB12 VITC SODIUM POTASIUM IRON
   9  0                 CALCIUM PHOSPHOR SEX
  10  0
  11  0  COMMENT ADD THE CALCULATED MISSING MEAL AVERAGES
  12  0

File DISK$USER2:[NUTRITION.T8805.STAT]SUPER.SYS;
   Created:  27-OCT-88 09:17:34 - 42 variables

  13  0  ADD FILES FILE=*/FILE=SYSFLE/MAP
  14  0

File DISK$USER2:[NUTRITION.T8805.STAT]MISSING.SYS;
   Created:   8-NOV-88 09:25:44 - 24 variables

>Note # 5151
>Information specific to SPSS-X Data Entry (for example, forms, ranges, and
>rules) will be lost, unless it is contained in the first file specified.
```

```
Map of the result file

Result     Input1    Input2      Result     Input1    Input2      Result     Input1    Input2
------     ------    ------      ------     ------    ------      ------     ------    ------
DATE       DATE      DATE        CHOLESTR   CHOLESTR  CHOLESTR    SODIUM     SODIUM    SODIUM
SUB        SUB       SUB         CARBOHYD   CARBOHYD  CARBOHYD    POTASIUM   POTASIUM  POTASIUM
MEAL       MEAL      MEAL        VITA       VITA      VITA        IRON       IRON      IRON
CODE       CODE      CODE        THIAMIN    THIAMIN   THIAMIN     CALCIUM    CALCIUM   CALCIUM
LOCAT      LOCAT     LOCAT       RIBOFLAV   RIBOFLAV  RIBOFLAV    PHOSPHOR   PHOSPHOR  PHOSPHOR
ENERGY     ENERGY    ENERGY      NIACIN     NIACIN    NIACIN      SEX        SEX       SEX
PROTEIN    PROTEIN   PROTEIN     VITB12     VITB12    VITB12      HEIGHT               HEIGHT
FAT        FAT       FAT         VITC       VITC      VITC        WT1                  WT1

  15  0  SELECT IF ((DATE NE 080188) AND (CODE NE 3))
  16  0
  17  0  SORT CASES BY SEX DATE SUB
  18  0
  19  0  COMMENT THE THREE LINES BELOW CALCULATE EACH SUBJECT'S
  20  0          DAILY NUTRIENT INTAKE.  THIS OPERATION USES THE
  21  0          AGGREGATE COMMAND.  THE COMMAND SUMS ACROSS THE
  22  0          VALUES FOR ENERGY, PROTEIN, FAT AND CARBOHYDRATES
  23  0          FOR EACH SUBJECT FOR EACH DATE.  THE RESULTS ARE
  24  0          NOT SAVED IN A NEW OUTFILE (AN OPTION), HENCE THE
  25  0          ASTERIX.  THE NEW, AGGREGATED INFORMATION IS STORED
  26  0          IN NEW VARIABLES NAMED SNRG, SPRO, SFAT AND SCAR.
  27  0
SIZE OF FILE TO BE SORTED:     14114 CASES OF    192 BYTES EACH.
SORT COMPLETED SUCCESSFULLY.  FILE SIZE:       5293 BLOCKS.

Preceding task required 130.10 seconds CPU time;  239.18 seconds elapsed.

  28  0  AGGREGATE   OUTFILE = */
  29  0            BREAK = SEX DATE SUB/
  30  0            SNRG SPRO SFAT SCAR = SUM(ENERGY PROTEIN FAT CARBOHYD)
  31  0

AGGREGATE problem requires 488 bytes of memory
plus 108 bytes per case in the aggregated file.

A new (aggregated) active file has replaced the existing active file.
It contains 7 variable(s) and 567 case(s).

Preceding task required 51.68 seconds CPU time;  114.04 seconds elapsed.

  32  0  VARIABLE LABELS
  33  0       SNRG "ENERGY"
  34  0       SPRO "PROTEIN"
  35  0       SFAT "FAT"
  36  0       SCAR "CARBOHYD"
  37  0
  38  0  VALUE LABELS
  39  0       SEX 1    "MALE"
  40  0          2     "FEMALE"/
  41  0
  42  0  COMMENT PERFORM ANOVAS ON DAILY NUTRIENT INTAKES BY SEX
  43  0
  44  0  ANOVA    SNRG BY SEX(1,2)
  45  0  STATISTICS 3
  46  0

ANOVA problem requires 145 bytes of memory.
```

```
26-Sep-89    DAILY MEAN INTAKE USING MISSING DATA.                                Page   4
14:41:32     USARIEM                          on VAX780::          VMS V4.7

File:      AGGREGATED FILE
                                 * * *  C E L L   M E A N S  * * *
                SNRG       ENERGY
            BY SEX         Sex

TOTAL POPULATION
  2837.85
 (    567)

SEX
          1          2
  3199.45    2467.21
 (    287)  (    280)

            * * *  A N A L Y S I S   O F   V A R I A N C E  * * *

                  SNRG       ENERGY
            by    SEX        Sex

                                      Sum of                   Mean              Sig
Source of Variation                  Squares      DF          Square         F   of F

Main Effects                        75990861       1  75990861.122   177.077   .000
   SEX                              75990861       1  75990861.122   177.077   .000

Explained                           75990861       1  75990861.122   177.077   .000

Residual                           242463995     565    429139.813

Total                              318454856     566    562641.088

567 cases were processed.
0 cases (.0 pct) were missing.
```

```
Preceding task required 1.93 seconds CPU time;  7.45 seconds elapsed.

  47  0  ANOVA   SPRO BY SEX(1,2)
  48  0  STATISTICS 3
  49  0

ANOVA problem requires 145 bytes of memory.
```

```
File:     AGGREGATED FILE
                        * * *  C E L L   M E A N S  * * *
                SPRO       PROTEIN
            BY SEX         Sex

TOTAL POPULATION
   110.92
 (    567)

SEX
        1          2
   125.52      95.95
 (    287)  (    280)

            * * *  A N A L Y S I S   O F   V A R I A N C E  * * *

                SPRO       PROTEIN
            by  SEX        Sex

                                         Sum of               Mean                 Sig
Source of Variation                     Squares       DF      Square          F    of F

Main Effects                         123854.502        1  123854.502    160.962    .000
   SEX                               123854.502        1  123854.502    160.962    .000

Explained                            123854.502        1  123854.502    160.962    .000

Residual                             434747.032      565     769.464

Total                                558601.534      566     986.929

567 cases were processed.
0 cases (.0 pct) were missing.
```

```
Preceding task required 1.65 seconds CPU time;  5.64 seconds elapsed.

  50  0  ANOVA   SFAT BY SEX(1,2)
  51  0  STATISTICS 3
  52  0

ANOVA problem requires 145 bytes of memory.
```

```
File:      AGGREGATED FILE
                      * * *  C E L L   M E A N S  * * *
              SFAT       FAT
           BY SEX        Sex

TOTAL POPULATION
   107.69
 (   567)

SEX
        1          2
   121.07      93.97
 (   287)  (   280)

            * * *  A N A L Y S I S   O F   V A R I A N C E  * * *

                SFAT      FAT
           by   SEX       Sex

                                             Sum of                 Mean                 Sig
Source of Variation                         Squares      DF       Square          F     of F

Main Effects                             104057.372       1   104057.372     74.182     .000
   SEX                                   104057.372       1   104057.372     74.182     .000

Explained                                104057.372       1   104057.372     74.182     .000

Residual                                 792545.368     565     1402.735

Total                                    896602.740     566     1584.104

567 cases were processed.
0 cases (.0 pct) were missing.
```

```
Preceding task required 1.71 seconds CPU time;  7.69 seconds elapsed.

  53  0  ANOVA   SCAR BY SEX(1,2)
  54  0  STATISTICS 3
  55  0
  56  0
  57  0

ANOVA problem requires 145 bytes of memory.
```

```
26-Sep-89   DAILY MEAN INTAKE USING M.SSING DATA.                                        Page 10
14:41:55    USARIEM                         on VAX780::          VMS V4.7

File:      AGGREGATED FILE
                       * * *  C E L L   M E A N S  * * *
                SCAR       CARBOHYD
             BY SEX        Sex

TOTAL POPULATION
   364.72
 (   567)

SEX
        1         2
   409.90    318.41
 (   287)  (   280)

             * * *  A N A L Y S I S   O F   V A R I A N C E  * * *

                 SCAR      CARBOHYD
             by  SEX       Sex

                                        Sum of                    Mean               Sig
Source of Variation                    Squares      DF          Square          F    of F

Main Effects                       1186154.596       1     1186154.596    157.361    .000
   SEX                             1186154.596       1     1186154.596    157.361    .000

Explained                          1186154.596       1     1186154.596    157.361    .000

Residual                           4258863.976     565        7537.812

Total                              5445018.572     566        9620.174

567 cases were processed.
0 cases (.0 pct) were missing.
```

```
26-Sep-89   DAILY MEAN INTAKE USING MISSING DATA.                                   Page  11
14:41:57    USARIEM                          on VAX780::           VMS V4.7

Preceding task required 1.65 seconds CPU time;  10.52 seconds elapsed.

  58  0  FINISH

          58 command lines read.
           0 errors detected.
           0 warnings issued.
         274 seconds CPU time.
         577 seconds elapsed time.
       End of job.
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                         Page    1
14:18:48    USARIEM                           on VAX780::          VMS V4.7

VAX-11/780               USARIEM                              License Number 17731
This software is functional through September 30, 1989.

Try the new SPSS-X Release 3.0 and 3.1 features:

* Interactive SPSS-X command execution          * The new RANK procedure
* Online,VMS-like Help                          * Improvements in:
* Nonlinear Regression                          *   REPORT and TABLES
* Time Series and Forecasting (TRENDS)          *   Simplified Syntax
* Macro Facility                                *   Matrix I/O

See SPSS-X User's Guide, Third Edition, for more information on these features.

   1  0
   2  0
   3  0  FILE HANDLE INFILE/NAME = '[NUTRITION.T8805.STAT]SUPER.SYS'
   4  0
   5  0  COMMENT  THIS OUTPUT IS AN EXAMPLE OF A FREQUENCIES COUNT
   6  0           FOR ALL FOODS CHOSEN BY SUBJECTS IN A GARRISON
   7  0           DINING FACILITY STUDY IN ORDER TO GET AN IDEA
   8  0           OF THE MOST POPULAR FOOD ITEMS.
   9  0
  10  0  GET FILES = INFILE/map
  11  0
  12  0

File DISK$USER2:[NUTRITION.T8805.STAT]SUPER.SYS;
   Created:  27-OCT-88 09:17:34 - 42 variables

FILE MAP

Result     Input1     Result     Input1     Result     Input1     Result     Input1     Result     Input1
------     ------     ------     ------     ------     ------     ------     ------     ------     ------
DATE       DATE       LOCAT      LOCAT      THIAMIN    THIAMIN    PHOSPHOR   PHOSPHOR   WT4        WT4
SUB        SUB        FOODCODE   FOODCODE   RIBOFLAV   RIBOFLAV   AGE        AGE        LSU        LSU
MEAL       MEAL       QUANTITY   QUANTITY   NIACIN     NIACIN     SEX        SEX        TRIG       TRIG
CODE       CODE       ENERGY     ENERGY     VITB12     VITB12     RACE       RACE       VLDL       VLDL
GRCODE     GRCODE     PROTEIN    PROTEIN    VITC       VITC       HEIGHT     HEIGHT     LDL        LDL
AMTEAT     AMTEAT     FAT        FAT        SODIUM     SODIUM     KODAC      KODAC      HDL        HDL
AMTRET     AMTRET     CHOLESTR   CHOLESTR   POTASIUM   POTASIUM   WT1        WT1
REASNOT    REASNOT    CARBOHYD   CARBOHYD   IRON       IRON       WT2        WT2
SALT       SALT       VITA       VITA       CALCIUM    CALCIUM    WT3        WT3

  13  0  FRENQUENCIES VARIABLES=FOODCODE/
  14  0              FORMAT=DFREQ/
  15  0
  16  0  COMMENT THE TWO COMMAND LINES ABOVE TELL THE COMPUTER TO PERFORM THE
  17  0          FREQUENCIES COUNT ON THE VARIABLE FOODCODE, AND TO PRODUCE
  18  0          OUTPUT THAT ORDERS THE FOODCODES IN DECREASING FREQUENCY
  19  0          ACCORDING TO THE NUMBER OF TIMES THAT PARTICULAR ITEM WAS
  20  0          CHOSEN.
  21  0

There are 466,656 bytes of memory available.

Memory allows a total of 15,917 values accumulated across all variables.
There may be up to 3,980 value labels for each variable.
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                               Page   2
14:19:41    USARIEM                          on VAX780::          VMS V4.7

FOODCODE  Food Code

                                                                Valid      Cum
  Value Label                   Value   Frequency   Percent   Percent   Percent

KOOLAID                        KOOLS         852       5.6       5.6       5.6
BREAD-MIX GRAIN                863007        743       4.9       4.9      10.5
MARGARINE, SOY                 804080        727       4.8       4.8      15.3
LETTUCE ICEBERG RAW            811252        536       3.5       3.5      18.8
BREAD-WHITE                    004610        441       2.9       2.9      21.7
MILK-LOWFAT 2%                 801079        401       2.6       2.6      24.4
BANANA MINUS SKIN              809040        386       2.5       2.5      26.9
MAPLE SYRUP-ARTIFICI           020520        375       2.5       2.5      29.4
ORANGE JUICE                   809209        350       2.3       2.3      31.7
BACON-COOKED                   810124        328       2.2       2.2      33.8
TOMATO-RAW                     811883        309       2.0       2.0      35.9
EGG-SCRAMBLED                  801132        303       2.0       2.0      37.9
CHEDDAR CHEESE                 801009        280       1.8       1.8      39.7
APPLE W  SKIN                  809003        245       1.6       1.6      41.3
EGG, HARD, CHOPPED             801129        228       1.5       1.5      42.8
SALT                           019630        219       1.4       1.4      44.3
PANCAKE                        D02550        216       1.4       1.4      45.7
MILK 2% CHOC                   801103        197       1.3       1.3      47.0
RICE 8  2L, 8  3D, 8           E00504        181       1.2       1.2      48.2
CORN, WK PLAIN 8  2L           QG0650        170       1.1       1.1      49.3
SALAD DRESSING-ITALI           804114        161       1.1       1.1      50.4
PEAR-FRESH-9%                  809252        159       1.0       1.0      51.4
PEACH-FRESH-13%                809236        150       1.0       1.0      52.4
VEAL PATTIE BREADED            U09201        150       1.0       1.0      53.4
CHOW MEIN NOODLES              013810        147       1.0       1.0      54.4
SUGAR                          ARC006        144        .9        .9      55.3
TOAST-MIX GRAIN WHEA           863008        143        .9        .9      56.3
GRAPE JUICE                    809137        142        .9        .9      57.2
TOAST                          004620        133        .9        .9      58.1
CARROTS, PLAIN, 8  1           QG0640        116        .8        .8      58.8
CEREAL-CORN FLAKES K           808020        113        .7        .7      59.6
ORANGE MINUS SKIN              809203        111        .7        .7      60.3
CUCUMBER-RAW                   811205        110        .7        .7      61.0
CRACKERS, SALTINES             SALTIN        102        .7        .7      61.7
WAFFLE, PLAIN, COMME           WAFFLE         97        .6        .6      62.3
SAUSAGE PATTY                  L08920         96        .6        .6      63.0
JELLO W  FRUIT COCKT           JELSD1         95        .6        .6      63.6
PEANUT BUTTER                  014990         94        .6        .6      64.2
WATER                          027010         91        .6        .6      64.8
SALAD DRESSING-FRENC           804120         91        .6        .6      65.4
FRENCH FRIES                   FRFRY          90        .6        .6      66.0
SALAD DRESSING-THOUS           804017         89        .6        .6      66.6
FRENCH TOAST 8  2              D022W          89        .6        .6      67.2
CEREAL-RICE KRISPIES           808065         81        .5        .5      67.7
RAISIN BRAN KELLG              808060         70        .5        .5      68.2
CATSUP                         022861         69        .5        .5      68.6
JELLY                          011491         63        .4        .4      69.0
MIXED VEGETABLES 8             QG1680         63        .4        .4      69.5
FR FR FISH PORTION             FISHPT         61        .4        .4      69.9
HAM STEAK                      L07110         60        .4        .4      70.3
TEA BREWED                     TEABAG         59        .4        .4      70.6
BROWN GRAVY 8  8D              001609         58        .4        .4      71.0
BROWN GRAVY 8  9D              001605         57        .4        .4      71.4
RICE PILAF 8  10L              E00801         54        .4        .4      71.8
BROWN GRAVY 8  9L              001604         54        .4        .4      72.1
GRAVY 8  3 L                   GRAVYA         53        .3        .3      72.5
RICE PILAF 8  1 D              E00800         51        .3        .3      72.8
POLISH SAUSAGE                 POLIS1         50        .3        .3      73.1
YOGURT, W  FRUIT YUM           801121         49        .3        .3      73.4
BRAISED PORK CHOPS 8           L08500         49        .3        .3      73.8
OATMEAL                        OATMEL         49        .3        .3      74.1
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                        Page    3
14:19:42    USARIEM                        on VAX780::           VMS V4.7

FOODCODE  Food Code

BEEF STEW 8  4L          L02200        47        .3        .3       74.4
PORK ADOBO               L09900        47        .3        .3       74.7
FRIED RICE 8  2 D        E00702        46        .3        .3       75.0
VEAL PARMESAN SAUCE      L1031S        46        .3        .3       75.3
JAM                      011482        45        .3        .3       75.6
BROWN GRAVY 8  3D        001608        45        .3        .3       75.9
GRILLED STEAK 8  9       STEAK         45        .3        .3       76.2
RICE 8  9D               E00507        44        .3        .3       76.5
BEEF STEW 8  2           STEWBF        44        .3        .3       76.8
CHILI 8  8L              CHILI         42        .3        .3       77.1
GRAPE                    809131        41        .3        .3       77.3
CAKE, SPONGE W  O FR     CAKESP        41        .3        .3       77.6
STEAMED RICE 8  5 L      E00502        41        .3        .3       77.9
HERBED GREEN BEANS 8     Q08200        41        .3        .3       78.1
SPANISH BEEF             SPBFP         41        .3        .3       78.4
NOODLES 8  5D            E00403        40        .3        .3       78.7
FRUIT COCKTAIL, CANN     FRCOCK        40        .3        .3       78.9
SWISS STEAK 8  3L        L01610        40        .3        .3       79.2
ROAST TURKEY 8  8D       L16202        40        .3        .3       79.5
BROWN GRAVY 8  5 D       001603        40        .3        .3       79.7
NOODLES 8  3             E01200        39        .3        .3       80.0
PEAS W  MUSHROOMS 8      Q04100        39        .3        .3       80.2
ROAST TURKEY 8  3L       L16200        38        .3        .3       80.5
POTATOES, PLAIN 8  8     Q07722        38        .3        .3       80.7
BROWNIE, FROSTED         BROWNI        37        .2        .2       81.0
BRAISED PORK CHOP 8      CHOP1         37        .2        .2       81.2
MASHED POTATOES 8  9     Q05703        36        .2        .2       81.5
BEANS, GREEN, CANNED     811056        35        .2        .2       81.7
RICE 8  8D               E00505        35        .2        .2       81.9
POT ROAST 8  8D          L00930        35        .2        .2       82.2
SPINICH-STEAMED NO S     811464        34        .2        .2       82.4
NOODLES 8  8D            E00402        34        .2        .2       82.6
GRITS 8  4               GRIT1         34        .2        .2       82.8
SWISS STEAK W  MSHRM     L01640        34        .2        .2       83.1
OVEN BROWNED POTATOE     Q05001        34        .2        .2       83.3
MASHED POTATOES 8  5     Q05702        34        .2        .2       83.5
CAULIFLOWER 8  2,8       QG0663        34        .2        .2       83.7
TURKEY ALA KING 8  5     TURK1         34        .2        .2       84.0
BROWN GRAVY 8  2L        001607        33        .2        .2       84.2
MASHED POTATOES 8  3     Q05708        33        .2        .2       84.4
HONEY                    011340        32        .2        .2       84.6
APPLESAUCE               809020        32        .2        .2       84.8
COFFEE BREWED            COFFEE        32        .2        .2       85.0
ROAST TURKEY 8  10D      L16205        32        .2        .2       85.2
TARTAR SAUCE             022730        31        .2        .2       85.4
BOLOGNA                  807008        31        .2        .2       85.6
C,  'AGE, COOKED, PLA    811110        31        .2        .2       85.8
GRAVY, BROWN 8  5 L      L0164G        31        .2        .2       86.0
MASHED POTATOES 8  5     Q05709        31        .2        .2       86.2
CAKE, BANANA             CAKEBN        30        .2        .2       86.4
JELLO W  PINEAPPLE       JELSD9        30        .2        .2       86.6
RICE 8  9L               E00503        29        .2        .2       86.8
LASAGNA 8  4D            L02500        29        .2        .2       87.0
PICKLE SWEET             015610        28        .2        .2       87.2
MILK-SKIM                801085        28        .2        .2       87.4
PLUM-FRESH-9%            809279        28        .2        .2       87.6
PEAS, CANNED, REG, D     811308        28        .2        .2       87.8
NOODLES 8  10D           E00404        28        .2        .2       87.9
BRAISED BEEF 8  10       L01712        28        .2        .2       88.1
MACARONI SLD 8  1 AV     M03401        28        .2        .2       88.3
SALAD DRESSING-FRENC     804020        27        .2        .2       88.5
CAKE, COCONUT            CAKECO        27        .2        .2       88.7
SEASONED GREEN BEANS     Q00510        27        .2        .2       88.9
NECTARINE                809191        26        .2        .2       89.0
SALISBURY STEAK 8  1     L03700        26        .2        .2       89.2
```

```
FOODCODE  Food Code

BROWN GRAVY 8  1L           001681         26        .2        .2      89.4
OVEN BR POTATO 8  10        Q05003         26        .2        .2      89.5
GREEN BEANS, PLAIN          QG063          26        .2        .2      89.7
SOUTHERN BAKED FISH         L11940         25        .2        .2      89.9
BROWN GRAVY W  BEEF         001606         25        .2        .2      90.0
MASHED POTATOES 8  1        Q05705         25        .2        .2      90.2
CARROT CAKE W  NUTS         CARCAK         24        .2        .2      90.4
NE BOILED CORNED BEE        L11110         24        .2        .2      90.5
ROAST TURKEY 8  1L          L16201         24        .2        .2      90.7
MACARONI SALAD 8  2         M03402         24        .2        .2      90.8
CHIX GRAVY 8  10            001682         24        .2        .2      91.0
                            XXXXXX         24        .2        .2      91.1
FRENCH TOAST 8  10          D02201         23        .2        .2      91.3
BAKED GLAZED HAM 8          HAMBAK         23        .2        .2      91.5
MASHED POTATOES, AVE        Q05704         23        .2        .2      91.6
MUSTARD GREENS, STEA        QG1610         23        .2        .2      91.8
PEAS AND CARROTS STE        811323         22        .1        .1      91.9
CAKE, GERMAN CHOCOLA        CAKEGC         22        .1        .1      92.0
LYONNAISE WAX BEAN N        Q0070B         22        .1        .1      92.2
MASHED POTATO 8  1D         Q05701         22        .1        .1      92.3
BROCCOLI STEAMED            811095         21        .1        .1      92.5
MASHED POTATO 8  2L         Q05706         21        .1        .1      92.6
MASHED POTATOES 8  3        Q05707         21        .1        .1      92.7
BROWN GRAVY 8  1D           001601         20        .1        .1      92.9
QUICK GRITS  8  3B          GRIT4          19        .1        .1      93.0
GRITS 8  9                  GRIT5          19        .1        .1      93.1
SAVORY CHIX BREAST Q        L158BQ         19        .1        .1      93.3
CARROT RAISIN SALAD         M005R          19        .1        .1      93.4
CRACKERS                    MR6701         19        .1        .1      93.5
ASPARAGUS, STEAMED          QG0621         19        .1        .1      93.6
SAUERKRAUT                  SAUERK         19        .1        .1      93.8
BREAD-AMERICAN RYE          004540         18        .1        .1      93.9
POTATO BAKED                811674         18        .1        .1      94.0
HOT GRITS 8  10             GRIT2          18        .1        .1      94.1
ROAST BEEF 8  1L            L00910         18        .1        .1      94.2
CHIX QUART BREAST 8         L1430B         18        .1        .1      94.3
MACARONI SALAD 8  5D        M03405         18        .1        .1      94.5
GRITS 8  2B                 808631         17        .1        .1      94.6
SAVORY CHIX LEG QTRS        L158LQ         17        .1        .1      94.7
MASHED POTATOES 8  1        POTATO         17        .1        .1      94.8
BEANS LIMA, CANNED          QG1650         17        .1        .1      94.9
PICKLE-DILL                 015580         16        .1        .1      95.0
CHEESE-COTTAGE              801012         16        .1        .1      95.1
GRITS 8  8                  GRIT3          16        .1        .1      95.2
MUSTARD-YELLOW              013730         15        .1        .1      95.3
GRAVY FOR SPAN BEEF         GRSPBF         15        .1        .1      95.4
YOGURT, TRIMLINE            LFYOG          15        .1        .1      95.5
OATMEAL 8  4                OAT1           15        .1        .1      95.6
CHIX QUART LEG 8  9L        L1430L         14        .1        .1      95.7
HARVARD BEETS 8  4L         Q00800         14        .1        .1      95.8
CAKE, CARROT                CAKECR         13        .1        .1      95.9
KOOLADE SWEET 8  9L         KOOL1          13        .1        .1      96.0
MACARONI SALAD 8  9D        M03407         13        .1        .1      96.1
MACARONI SALAD 8  3D        M03409         13        .1        .1      96.2
POTATO SALAD 8  4L          M04004         13        .1        .1      96.2
POTATO SALAD 8  9D          M04010         13        .1        .1      96.3
THREE BEAN SALAD 8          M04501         13        .1        .1      96.4
BROWN GRAVY 8  4 L          001602         13        .1        .1      96.5
RISSOLE POTATOES            Q05200         13        .1        .1      96.6
VEGETABLES FOR NE DI        L11115         12        .1        .1      96.7
MACARONI SALAD 8  10        M03708         12        .1        .1      96.7
STEAK SAUCE                 STKSAU         12        .1        .1      96.8
DIET COLA                   02404D         11        .1        .1      96.9
PEACH DICED CND             014830         11        .1        .1      97.0
BEEF & NOODLES 8  10        BFNOOD         11        .1        .1      97.0
```

```
FOODCODE  Food Code

CARROT RAISIN SALAD      M005R1        11       .1       .1      97.1
COLESLAW 8  1L, 8  3     M00901        11       .1       .1      97.2
COLESLAW, AVE RX         M00902        11       .1       .1      97.3
POTATO SALAD 8  5D       M04005        11       .1       .1      97.3
GUM-NO NUTRIENTS         MR6816        11       .1       .1      97.4
PEARS, LIGHT SIRUP P .   809256        10       .1       .1      97.5
JELLO W  PEACHES 8       JELSD2        10       .1       .1      97.5
COLESLAW 8  3L           M00903        10       .1       .1      97.6
COLESLAW 8  1D,8  2L     M00909        10       .1       .1      97.7
COTTAGE CHEESE W  PI     M018PA        10       .1       .1      97.7
POTATO SALAD 8  2        M04002        10       .1       .1      97.8
COTTAGE CHEESE W  PE     PECOTC        10       .1       .1      97.9
BRUSSEL SPROUTS STEA     811101         9       .1       .1      97.9
COLESLAW 8  8D           M00906         9       .1       .1      98.0
CHEESE SPREAD            MR6901         9       .1       .1      98.0
MACARONI SLD 8  8 D      M03406         8       .1       .1      98.1
POTATO SALAD 8  1D       M04009         8       .1       .1      98.1
CARROTS NORMANDIE 8      Q01731         8       .1       .1      98.2
BU POTATOES 8  1L        Q04900         8       .1       .1      98.2
SEASONED GREEN BEANS     QG1630         8       .1       .1      98.3
SWEET AND LO             SWEELO         8       .1       .1      98.4
TABASCO SAUCE            TABASC         8       .1       .1      98.4
MUSHROOMS, CANNED        811264         7       .0       .0      98.5
BU NOODLES 8  1          E00401         7       .0       .0      98.5
FRUIT MIX                FRMIX          7       .0       .0      98.5
POTATO SALAD 8  3L       M04003         7       .0       .0      98.6
COCOA POWDER-RECONST     MR60X1         7       .0       .0      98.6
PEACHES FREEZE DRIED     MR6603         7       .0       .0      98.7
HASH BR POTATOES 8       Q0462B         7       .0       .0      98.7
JUICE GRAPE CND          809135         6       .0       .0      98.8
COLESLAW 8  9D           M00910         6       .0       .0      98.8
MACARONI SALAD 8  3L     M03403         6       .0       .0      98.8
POTATO SALAD 8  8L       M04006         6       .0       .0      98.9
BEANS IN TOMATO SAUC     MR6313         6       .0       .0      98.9
COOKIES-CHOC COV         MR6801         6       .0       .0      99.0
PEANUT BUTTER MRE &      MR6906         6       .0       .0      99.0
SOUR CREAM               801056         5       .0       .0      99.0
PINEAPPLE CANNED IN      809268         5       .0       .0      99.1
CARROT RAISIN SLD 8      M005R2         5       .0       .0      99.1
CANDY-AVG MRE            MR6815         5       .0       .0      99.1
JELLY MRE                MR6903         5       .0       .0      99.2
LYONNAISE WAX BEANS      Q00700   [illegible]   .0       .0      99.2
CRANBERRY SAUCE          809081   [illegible]   .0       .0      99.2
ONION WHITE RAW          811282         4       .0       .0      99.3
EGG WHITE NO YOLK        EGGWHI         4       .0       .0      99.3
CHEESE OMELET            F00830         4       .0       .0      99.3
PLUMS, CANNED, HEAVY     FRPLUM         4       .0       .0      99.3
CARROT RAISIN SALAD      M005R3         4       .0       .0      99.4
HAM  CHICKEN LOAF        MP6302         4       .0       .0      99.4
APPLESAUCE               MR6601         4       .0       .0      99.4
ONION FOR LO SWISS S     ONION          4       .0       .0      99.4
SPICE CAKE               SPICEC         4       .0       .0      99.5
OLIVE-BLACK              014090         3       .0       .0      99.5
JUICE, PINEAPPLE, CA     809409         3       .0       .0      99.5
CAKE, APPLE CINNAMON     CAKEAC         3       .0       .0      99.5
RICE 8  5D               E00501         3       .0       .0      99.5
EGG YOLK                 EGGYLK         3       .0       .0      99.6
COLESLAW 8  10           M00908         3       .0       .0      99.6
FRANKFURTER              MR6306         3       .0       .0      99.6
BROWNIE-CHOCOLATE CO     MR6803         3       .0       .0      99.6
CAKE-CHERRY NUT          MR6804         3       .0       .0      99.6
COOKIES, OREO            OREO           3       .0       .0      99.7
SODA FRUIT               004060         2       .0       .0      99.7
OLIVE-GREEN              014060         2       .0       .0      99.7
APPLE JUICE CANNED       809016         2       .0       .0      99.7
```

```
26-Sep-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                                      Page    6
14:19:44    USARIEM                          on VAX780::           VMS V4.7

FOODCODE  Food Code

COFFEE-INSTANT RECON       MR60X2          2         .0         .0       99.7
PORK SAUSAGE PATTY         MR6301          2         .0         .0       99.7
BEEF PATTY                 MR6303          2         .0         .0       99.7
BEEF STEW                  MR6305          2         .0         .0       99.8
BEEF W   GRAVY             MR6308          2         .0         .0       99.8
MEATBALLS W   BARBEQ       MR6310          2         .0         .0       99.8
HAM SLICES                 MR6311          2         .0         .0       99.8
POTATO PATTY NO!           MR6401          2         .0         .0       99.8
FRUIT MIX DEHYDRATED       MR6604          2         .0         .0       99.8
CAKE-CHOCOLATE NUT N       MR6807          2         .0         .0       99.8
CAKE-ORANGE NUT ROLL       MR6808          2         .0         .0       99.8
CARBONATED, ROOT BEE       004080          1         .0         .0       99.9
BLUEBERRY PIE              015697          1         .0         .0       99.9
CREAMER, NON DAIRY,        801067          1         .0         .0       99.9
COCOA HOT                  801105          1         .0         .0       99.9
FRANKFURTER                807023          1         .0         .0       99.9
HAMBURGER  HOTDOG RL       863041          1         .0         .0       99.9
CANDY, CARAMEL             CARAME          1         .0         .0       99.9
CHEESECAKE SARA LEE        CHCAKE          1         .0         .0       99.9
COOKIE, CHOCO CHIP,        CHOCCH          1         .0         .0       99.9
OMELET, PLAIN              F00800          1         .0         .0       99.9
COOKIE, DOUBLE FUDGE       FDGCOO          1         .0         .0       99.9
APRICOTS, CANNED, LI       FRAPR           1         .0         .0       99.9
ICE CREAM SANDWICH         ICESAN          1         .0         .0       99.9
CREAMED BEEF 8  4          L03004          1         .0         .0       99.9
CREAMED BEEF 8  8          L03006          1         .0         .0       99.9
CRM GRD BEEF 8  10         L03008          1         .0         .0      100.0
THREE BEAN SALAD PER       M045            1         .0         .0      100.0
CREAM SUBSTITUTE-NON       MR6003          1         .0         .0      100.0
BEEF W   BARBEQUE SA       MR6304          1         .0         .0      100.0
CAKE-PINEAPPLE NUT N       MR6802          1         .0         .0      100.0
SOUP  GRAVY BASE MRE       MR6905          1         .0         .0      100.0
GRANULATED SUGAR           MR6909          1         .0         .0      100.0
CAULIFLOWER 8  2           QG0661          1         .0         .0      100.0
                                        -------    -------    -------
                             Total       15184      100.0      100.0

Valid cases    15184      Missing cases      0
```

```
26-Sep-89    SPSS-X RELEASE 3.1 FOR VAX/VMS                                            Page    7
14:19:45     USARIEM                          on VAX780::           VMS V4.7

Preceding task required 32.10 seconds CPU time;  54.19 seconds elapsed.

  22  0  FINISH

          22 command lines read.
           0 errors detected.
           0 warnings issued.
          35 seconds CPU time.
          58 seconds elapsed time.
       End of job.
```

```
26-SEP-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                       Page   1
14:08:01    USARIEM                        on VAX780::          VMS V4.7

VAX-11/780              USARIEM                          License Number 1773
This software is functional through September 30, 1990.

Try the new SPSS-X Release 3.0 and 3.1 features:

* Interactive SPSS-X command execution           * The new RANK procedure
* Online,VMS-like Help                           * Improvements in:
* Nonlinear Regression                           *   REPORT and TABLES
* Time Series and Forecasting (TRENDS)           *   Simplified Syntax
* Macro Facility                                 *   Matrix I/O

See SPSS-X User's Guide, Third Edition, for more information on these features.

   1  0
   2  0
   3  0  FILE HANDLE INFILE/NAME = '[NUTRITION.T8805.STAT]SUPER.SYS'
   4  0
   5  0  COMMENT THIS FILE IS AN EXAMPLE OF AN OUTPUT THAT IS VERY COMMON
   6  0          IN GARRISON DINING FACILITY STUDIES CONDUCTED AT USARIEM.
   7  0          RESEARCHERS MIGHT WANT TO KNOW, FOR EXAMPLE, THE FOODS TO WHICH
   8  0          SUBJECTS ADDED SALT.  FURTHERMORE, IT WOULD BE NICE TO HAVE
   9  0          THE COMPUTER PRINT OUT THESE FOODS ORGANIZED SO THAT THE
  10  0          FOOD THAT HAD SALT ADDED TO IT MOST FREQUENTLY WOULD BE FIRST,
  11  0          FOLLOWED BY THE OTHER FOODS IN DECREASING ORDER.  THE FOLLOWING
  12  0          INSTRUCTIONS PRODUCE SUCH AN OUTPUT.
  13  0
  14  0  GET FILES = INFILE/map
  15  0
  16  0

File DISK$USER2:[NUTRITION.T8805.STAT]SUPER.SYS;
   Created:  27-OCT-88 09:17:34 - 42 variables

FILE MAP

Result    Input1    Result    Input1    Result    Input1    Result    Input1    Result    Input1
------    ------    ------    ------    ------    ------    ------    ------    ------    ------
DATE      DATE      LOCAT     LOCAT     THIAMIN   THIAMIN   PHOSPHOR  PHOSPHOR  WT4       WT4
SUB       SUB       FOODCODE  FOODCODE  RIBOFLAV  RIBOFLAV  AGE       AGE       LSU       LSU
MEAL      MEAL      QUANTITY  QUANTITY  NIACIN    NIACIN    SEX       SEX       TRIG      TRIG
CODE      CODE      ENERGY    ENERGY    VITB12    VITB12    RACE      RACE      VLDL      VLDL
GRCODE    GRCODE    PROTEIN   PROTEIN   VITC      VITC      HEIGHT    HEIGHT    LDL       LDL
AMTEAT    AMTEAT    FAT       FAT       SODIUM    SODIUM    KODAC     KODAC     HDL       HDL
AMTRET    AMTRET    CHOLESTR  CHOLESTR  POTASIUM  POTASIUM  WT1       WT1
REASNOT   REASNOT   CARBOHYD  CARBOHYD  IRON      IRON      WT2       WT2
SALT      SALT      VITA      VITA      CALCIUM   CALCIUM   WT3       WT3

  17  0  SELECT IF ((DATE NE 080188) AND (CODE NE 3))
  18  0
  19  0  COMMENT THE ABOVE LINE DELETES ONE GROUP AND ONE DAY FROM THE
  20  0          ANALYSIS FOR INTERNAL REASONS HAVING NOTHING TO DO WITH
  21  0          THE EXAMPLE.  tHIS IS NOT A HYPOTHETICAL DATA SET, AND
  22  0          SUCH PROBLEMS OCCASIONALLY CROP UP.
  23  0
  24  0  SELECT IF (SALT = "Y")
  25  0
  26  0  COMMENT NOT THAT SINCE THE VARIABLE REPRESENTING WHETHER OR NOT
  27  0          SALT WAS ADDED TO A PARTICULAR FOOD (I.E., VARIABLE NAME SALT) WAS
  28  0          CODED AS A Y OR N, REPRESENTING YES AND NO RESPECTIVELY,
  29  0          THE CODE HAS TO BE ENCLOSED IN QUOTATION MARKS WHEN REFERED TO,
  30  0          AS IT WAS NOT CODED NUMERICALLY.  SUCH CODINGS ARE CALLED
  31  0          LITERALS; SEE THE SPSSX MANUAL FOR DEALING WITH THIS SITUATION.
  32  0
  33  0  FREQUENCIES VARIABLES=FOODCODE/
  34  0               FORMAT=DFREQ/
  35  0
  36  0  COMMENT THE TWO COMMAND LINES ABOVE TELL THE COMPUTER TO PERFORM THE
  37  0          FREQUENCIES COUNT ON THE VARIABLE FOODCODE, AND TO PRODUCE
  38  0          OUTPUT THAT ORDERS THE FOODCODES IN DECREASING FREQUENCY
  39  0          ACCORDING TO THE NUMBER OF TIMES THAT SALT WAS ADDED TO A
  40  0          PARTICULAR ITEM.
  41  0

There are 4,442,944 bytes of memory available.

Memory allows a total of 32,767 values accumulated across all variables.
There may be up to 8,192 value labels for each variable.
```

```
FOODCODE  Food Code

                                                                  Valid      Cum
  Value Label                   Value  Frequency  Percent  Percent  Percent

EGG-SCRAMBLED                  801132        29      9.4      9.4      9.4
LETTUCE ICEBERG RAW            811252        13      4.2      4.2     13.6
VEAL PATTIE BREADED            U09201        10      3.2      3.2     16.8
EGG, HARD, CHOPPED             801129         9      2.9      2.9     19.7
FRENCH FRIES                   FRFRY          9      2.9      2.9     22.7
CORN, WK PLAIN 8  2L           QG0650         7      2.3      2.3     24.9
MIXED VEGETABLES 8             QG1680         7      2.3      2.3     27.2
RICE 8  9L                     E00503         6      1.9      1.9     29.1
RICE 8  2L, 8  3D, 8           E00504         6      1.9      1.9     31.1
POT ROAST 8  8D                L00930         6      1.9      1.9     33.0
SWISS STEAK 8  3L              L01610         6      1.9      1.9     35.0
OVEN BR POTATO 8  10           Q05003         6      1.9      1.9     36.9
MASHED POTATOES 8  9           Q05703         6      1.9      1.9     38.8
MASHED POTATOES 8  3           Q05708         6      1.9      1.9     40.8
SPINICH-STEAMED NO S           811464         5      1.6      1.6     42.4
POTATO BAKED                   811674         5      1.6      1.6     44.0
TOMATO-RAW                     811883         5      1.6      1.6     45.6
NOODLES 8  8D                  E00402         5      1.6      1.6     47.2
RICE 8  8D                     E00505         5      1.6      1.6     48.9
RICE 8  9D                     E00507         5      1.6      1.6     50.5
MASHED POTATOES, AVE           Q05704         5      1.6      1.6     52.1
MASHED POTATOES 8  1           Q05705         5      1.6      1.6     53.7
CARROTS, PLAIN, 8  1           QG0640         5      1.6      1.6     55.3
GRILLED STEAK 8  9             STEAK          5      1.6      1.6     57.0
CHEDDAR CHEESE                 801009         4      1.3      1.3     58.3
STEAMED RICE 8  5 L            E00502         4      1.3      1.3     59.5
RICE PILAF 8  10L              E00801         4      1.3      1.3     60.8
BRAISED BEEF 8  10             L01712         4      1.3      1.3     62.1
PORK ADOBO                     L09900         4      1.3      1.3     63.4
ROAST TURKEY 8  10D            L16205         4      1.3      1.3     64.7
MASHED POTATOES 8  5           Q05709         4      1.3      1.3     66.0
SPANISH BEEF                   SPBFP          4      1.3      1.3     67.3
CHOW MEIN NOODLES              013810         3      1.0      1.0     68.3
BROCCOLI STEAMED               811095         3      1.0      1.0     69.3
BEEF & NOODLES 8  10           BFNOOD         3      1.0      1.0     70.2
CHILI 8  8L                    CHILI          3      1.0      1.0     71.2
NOODLES 8  10D                 E00404         3      1.0      1.0     72.2
NOODLES 8  3                   E01200         3      1.0      1.0     73.1
QUICK GRITS  8  3B             GRIT4          3      1.0      1.0     74.1
NE BOILED CORNED BEE           L11110         3      1.0      1.0     75.1
CHIX QUART LEG 8  9L           L1430L         3      1.0      1.0     76.1
ROAST TURKEY 8  8D             L16202         3      1.0      1.0     77.0
OVEN BROWNED POTATOE           Q05001         3      1.0      1.0     78.0
BEEF STEW 8  2                 STEWBF         3      1.0      1.0     79.0
BACON-COOKED                   810124         2       .6       .6     79.6
BRAISED PORK CHOP 8            CHOP1          2       .6       .6     80.3
GRITS 8  8                     GRIT3          2       .6       .6     80.9
BEEF STEW 8  4L                L02200         2       .6       .6     81.6
BRAISED PORK CHOPS 8           L08500         2       .6       .6     82.2
ROAST TURKEY 8  3L             L16200         2       .6       .6     82.8
BROWN GRAVY 8  8D              001609         2       .6       .6     83.5
POLISH SAUSAGE                 POLIS1         2       .6       .6     84.1
HASH BR POTATOES 8             Q0462B         2       .6       .6     84.8
RISSOLE POTATOES               Q05200         2       .6       .6     85.4
MASHED POTATO 8  2L            Q05706         2       .6       .6     86.1
HERBED GREEN BEANS 8           Q08200         2       .6       .6     86.7
CAULIFLOWER 8  2,8             QG0663         2       .6       .6     87.4
TURKEY ALA KING 8  5           TURK1          2       .6       .6     88.0
PICKLE-DILL                    015580         1       .3       .3     88.3
YOGURT, W  FRUIT YUM           801121         1       .3       .3     88.7
SALAD DRESSING-ITALI           804114         1       .3       .3     89.0
```

```
26-SEP-89    SPSS-X RELEASE 3.1 FOR VAX/VMS                                                    Page    4
14:09:25     USARIEM                         on VAX780::          VMS V4.7

FOODCODE  Food Ccde

SALAD DRESSING-FRENC          804120          1        .3        .3      89.3
GRITS 8  2B                   808631          1        .3        .3      89.6
PEACH-FRESH-13%               809236          1        .3        .3      90.0
BEANS, GREEN, CANNED          811056          1        .3        .3      90.3
BRUSSEL SPROUTS STEA          811101          1        .3        .3      90.6
CABBAGE, COOKED, PLA          811110          1        .3        .3      90.9
NOODLES 8  5D                 E00403          1        .3        .3      91.3
FR FR FISH PORTION            FISHPT          1        .3        .3      91.6
GRITS 8  4                    GRIT1           1        .3        .3      91.9
HOT GRITS 8  10               GRIT2           1        .3        .3      92.2
GRITS 8  9                    GRIT5           1        .3        .3      92.6
SWISS STEAK W  MSHRM          L01640          1        .3        .3      92.9
LASAGNA 8  4D                 L02500          1        .3        .3      93.2
CREAMED BEEF 8  8             L03006          1        .3        .3      93.5
HAM STEAK                     L07110          1        .3        .3      93.9
SAUSAGE PATTY                 L08920          1        .3        .3      94.2
VEGETABLES FOR NE DI          L11115          1        .3        .3      94.5
SOUTHERN BAKED FISH           L11940          1        .3        .3      94.8
CHIX QUART BREAST 8           L1430B          1        .3        .3      95.1
SAVORY CHIX BREAST Q          L158BQ          1        .3        .3      95.5
COLESLAW, AVE RX              M00902          1        .3        .3      95.8
COLESLAW 8  3L                M00903          1        .3        .3      96.1
COLESLAW 8  8D                M00906          1        .3        .3      96.4
COLESLAW 8  9D                M00910          1        .3        .3      96.8
MACARONI SLD 8  1 AV          M03401          1        .3        .3      97.1
POTATO SALAD 8  9D            M04010          1        .3        .3      97.4
BROWN GRAVY 8  3D             001608          1        .3        .3      97.7
CHIX GRAVY 8  10              001682          1        .3        .3      98.1
SEASONED GREEN BEANS          Q00510          1        .3        .3      98.4
HARVARD BEETS 8  4L           Q00800          1        .3        .3      98.7
MASHED POTATOES 8  5          Q05702          1        .3        .3      99.0
POTATOES, PLAIN 8  8          Q07722          1        .3        .3      99.4
ASPARAGUS, STEAMED            QG0621          1        .3        .3      99.7
SAUERKRAUT                    SAUERK          1        .3        .3     100.0
                                          -------   -------   -------
                               Total         309     100.0     100.0

Valid cases      309      Missing cases      0
```

```
Preceding task required 50.46 seconds CPU time;  66.15 seconds elapsed.

  42  0  FINISH

          42 command lines read.
           0 errors detected.
           0 warnings issued.
          56 seconds CPU time.
          87 seconds elapsed time.
       End of job.
```

```
26-SEP-89   SPSS-X RELEASE 3.1 FOR VAX/VMS                                   Page   4
14:09:25    USARIEM                        on VAX780::          VMS V4.7

FOODCODE  Food Code

SALAD DRESSING-FRENC       804120          1        .3        .3      89.3
GRITS 8  2B                808631          1        .3        .3      89.6
PEACH-FRESH-13%            809236          1        .3        .3      90.0
BEANS, GREEN, CANNED       811056          1        .3        .3      90.3
BRUSSEL SPROUTS STEA       811101          1        .3        .3      90.6
CABBAGE, COOKED, PLA       811110          1        .3        .3      90.9
NOODLES 8  5D              E00403          1        .3        .3      91.3
FR FR FISH PORTION         FISHPT          1        .3        .3      91.6
GRITS 8  4                 GRIT1           1        .3        .3      91.9
HOT GRITS 8  10            GRIT2           1        .3        .3      92.2
GRITS 8  9                 GRIT5           1        .3        .3      92.6
SWISS STEAK W  MSHRM       L01640          1        .3        .3      92.9
LASAGNA 8  4D              L02500          1        .3        .3      93.2
CREAMED BEEF 8  8          L03006          1        .3        .3      93.5
HAM STEAK                  L07110          1        .3        .3      93.9
SAUSAGE PATTY              L08920          1        .3        .3      94.2
VEGETABLES FOR NE DI       L11115          1        .3        .3      94.5
SOUTHERN BAKED FISH        L11940          1        .3        .3      94.8
CHIX QUART BREAST 8        L1430B          1        .3        .3      95.1
SAVORY CHIX BREAST Q       L158BQ          1        .3        .3      95.5
COLESLAW, AVE RX           M00902          1        .3        .3      95.8
COLESLAW 8  3L             M00903          1        .3        .3      96.1
COLESLAW 8  8D             M00906          1        .3        .3      96.4
COLESLAW 8  9D             M00910          1        .3        .3      96.8
MACARONI SLD 8  1 AV       M03401          1        .3        .3      97.1
POTATO SALAD 8  9D         M04010          1        .3        .3      97.4
BROWN GRAVY 8  3D          O01608          1        .3        .3      97.7
CHIX GRAVY 8  10           O01682          1        .3        .3      98.1
SEASONED GREEN BEANS       Q00510          1        .3        .3      98.4
HARVARD BEETS 8  4L        Q00800          1        .3        .3      98.7
MASHED POTATOES 8  5       Q05702          1        .3        .3      99.0
POTATOES, PLAIN 8  8       Q07722          1        .3        .3      99.4
ASPARAGUS, STEAMED         QG0621          1        .3        .3      99.7
SAUERKRAUT                 SAUERK          1        .3        .3     100.0
                                       -------   -------   -------
                             Total         309     100.0     100.0

Valid cases      309      Missing cases      0
```

CHAPTER 7.0

MASTER FILE OF A RATION RECIPES

Doris Sherman

MASTER FILE OF A-RATION RECIPES

Objective:

The A ration recipes in the Master File will be the basis of nutritional analyses of Army cycle menus, diet histories, and other food consumption data for which the exact recipe used in food preparation is not available.

Major Points:

Following is a list of 1604 basic recipes found in Change 3 of the Armed Forces Recipe Service, TM-10-412 (AFRS). Each name is mutually exclusive and has been shortened to less than 70 characters for the file. This should be considered the bare bones of the names for the recipe file since the list will be expanded as recipes with optional ingredients are added to the list.

The columns are headed as follows.

Chg	This refers to the version of changes to the recipes that is currently in use. A blank indicated the recipe was in the original file (March 1980). Subsequent changes have been 1 (March 1984), Change 2 (March 1986) and Change 3 (May, 1987). The change number is found in the lower left hand corner of each recipe card.
Sec	One of the seventeen sections, A-Q, dividing the recipes. Each section is coded for a different group of recipes. For example, the C section contains Beverages. This alpha character is found in the upper right hand corner of each recipe card. As a subgroup, some sections of the recipes have guideline (G) cards which contain recipes. For example, DG6 contains guideline recipes for hot rolls.
No	This refers to the recipe number within each section. These numbers are not necessarily sequential. This number appears in the upper right hand corner of the recipe card after the Section.
Var	Some recipes contain variations to the main recipe. These are found on the same recipe card(s) with sequential variation numbers.
Recipe Name	The name of the recipe appears as it is presented at the top center of the recipe card or the name of the variation which is found after the variation number.

Future Plans:

It is expected that the coding of all of the A ration recipes will be completed by 1993. Until all of the recipes are coded, MND dietitians will have to code recipes for menu analyses as the need arises.

Armed Forces Recipe Service

Chg	Sec	No Var	Recipe Name
3	A	9	GUIDELINES FOR CONTAINER YIELDS FOR CND FRUITS
3	A	13	GUIDELINES FOR FRUIT BARS
2	B	1	CRANBERRY & ORANGE JCE COCKTAIL
2	B	1.01	CRANBERRY & APPLE JCE COCKTAIL
1	B	2	ORANGE FRUIT SHRUB
1	B	2.01	LEMON FRUIT SHRUB
3	B	3	TOMATO JCE COCKTAIL
3	B	3.01	VEGETABLE JCE COCKTAIL
3	B	3.02	SPICY TOMATO JCE COCKTAIL
1	B	4	SHRIMP COCKTAIL
3	B	5	NACHOS
3	B	6	PIZZA TREATS
2	CG	2	GUIDELINES FOR FRUIT & VEGETABLE JCES
1	CG	3	GUIDELINES FOR USE OF POWDERED BEVERAGE BASES
3	C	1	HOT COCOA
3	C	1.01	HOT WHIPPED COCOA
	C	2	COFFEE (Instnt Freeze-Dried)
1	C	3	COFFEE (Automatic Coffee Maker)
1	C	5	COFFEE (Automatic Urn)
1	C	5.01	COFFEE (Manual Urn)
3	C	6	FRUIT PUNCH
3	C	6.01	LIME LEMON PUNCH
3	C	7	ORANGE & PINEAPPLE JCE COCKTAIL
3	C	7.01	GRAPEFRUIT & PINEAPPLE JCE COCKTAIL
3	C	8	LEMONADE
3	C	8.01	LIMEADE
3	C	10	ORANGEADE
2	C	11	PINEAPPLEADE
1	C	12	HOT TEA
1	C	13	ICED TEA (Instnt)
1	C	13.01	ICED TEA (Instnt, for dispenser)
1	C	13.02	ICED TEA (Instnt, w/lemon & sugar, for dispenser)
1	C	13.03	ICED TEA (Tea, black, loose)
1	DG	6.01	CLOVERLEAF OR TWIN ROLLS (Hot Roll Make-up)
1	DG	6.02	DINNER OR (FINGER) FRANKFURTER ROLLS (Hot Roll Make-up)
1	DG	6.03	HAMBURGER OR SANDWICH ROLLS (Hot Roll Make-up)
1	DG	6.04	PAN, CLUSTER OR PULL APART ROLLS (Hot Roll Make-up)
1	DG	6.05	PARKER HOUSE ROLLS (Hot Roll Make-up)
1	DG	6.06	POPPY SEED OR SESAME SEED ROLLS (Hot Roll Make-up)
3	DG	7.01	GLAZED ROLLS (Sweet Dough Make-up)
3	DG	7.02	PRALINE ROLLS (Sweet Dough Make-up)
3	DG	7.03	CINNAMON ROLLS (Sweet Dough Make-up)
3	DG	7.04	BUTTERFLY ROLLS (Sweet Dough Make-up)
3	DG	7.05	PECAN ROLLS (Sweet Dough Make-up)
3	DG	7.06	SUGAR ROLLS (Sweet Dough Make-up)
3	DG	7.07	STREUSEL COFFEE CAKE (Sweet Dough Make-up)
3	DG	7.08	SMALL COFFEE CAKE (Sweet Dough Make-up)
3	DG	7.09	SWEDISH TEA RINGS (Sweet Dough Make-up)
3	DG	7.10	TWIST COFFEE CAKES (Sweet Dough Make-up)
3	DG	7.11	BEAR CLAWS (Sweet Dough Make-up)
3	DG	7.12	SNAILS (Sweet Dough Make-up)

3	DG	7.13	BOWKNOT, CHAIN TWIST, FIGURE 8, & S SHAPES (Sweet Dough Make-up)
3	DG	7.14	PLAIN TWISTS (Sweet Dough Make-up)
3	DG	7.15	BUTTERHORNS (Sweet Dough Make-up)
3	DG	7.16	CRESCENTS (Sweet Dough Make-up)
3	D	1	BAKING POWDER BISCUITS
3	D	1.01	BAKING POWDER BISCUITS (Biscuit Mix)
3	D	1.02	CHEESE BISCUITS
3	D	1.03	DROP BISCUITS
1	D	2	IRISH SODA BREAD
2	D	3	APPLE FILLING (Cnd Apples)
2	D	3.01	APPLE FILLING (Pie Filling, Prep)
2	D	3.02	APPLE FILLING (Dehy Apple Pie Filling Mix)
2	D	3.03	APPLE FILLING (Dehy Apple Pie Filling, Comp)
3	D	4	FRENCH BREAD
	D	5	RAISIN BREAD
	D	6	RYE BREAD
3	D	7	TOASTED GARLIC BREAD
3	D	7.01	TOASTED PARMESAN BREAD
3	D	7.02	TEXAS TOAST
2	D	8	WHITE BREAD
2	D	9	WHITE BREAD (Short-Time Formula)
2	D	9.01	RAISIN & MOLASSES BREAD (Short-Time Formula)
2	D	9.02	WHOLE WHEAT BREAD (Short-Time Formula)
	D	10	WHOLE WHEAT BREAD
3	D	11	PUMPKIN BREAD
1	D	12	CRUMB CAKE (Snickerdoodle)
1	D	12.01	CRUMB CAKE (Yellow Cake Mix)
3	D	13	CHINESE EGG ROLLS
3	D	13.01	FRIED CHINESE EGG ROLLS
3	D	13.02	LUMPIA EGG ROLLS
3	D	13.03	FRIED LUMPIA EGG ROLLS
2	D	14	CORN BREAD
2	D	14.01	CORN MUFFINS
2	D	14.02	HUSH PUPPIES
2	D	14.03	JALAPENO CORN BREAD
2	D	15	CORN BREAD (Corn Bread Mix)
2	D	15.01	CORN MUFFINS (Corn Bread Mix)
2	D	15.02	HUSH PUPPIES (Corn Bread Mix)
2	D	15.03	JALAPENO CORN BREAD (Corn Bread Mix)
1	D	16	CROUTONS
1	D	16.01	GARLIC CROUTONS
1	D	16.02	PARMESAN CROUTONS
	D	17	DANISH PASTRY
	D	17.01	DANISH PASTRY (Sweet Dough Mix)
3	D	18	CAKE DOUGHNUTS
3	D	18.01	SUGAR COATED DOUGHNUTS
3	D	18.02	CAKE DOUGHNUTS (Doughnut Mix)
3	D	18.03	CHOCOLATE DOUGHNUTS
3	D	18.04	CINNAMON SUGAR DOUGHNUTS
3	D	18.05	CRULLERS
3	D	18.06	GLAZED COCONUT DOUGHNUTS

3	D	18.07	GLAZED DOUGHNUTS
3	D	18.08	GLAZED NUT DOUGHNUTS
3	D	19	RAISED DOUGHNUTS
3	D	19.01	BEIGNETS (New Orleans Doughnuts)
3	D	19.02	RAISED DOUGHNUTS (Sweet Dough Mix)
3	D	19.03	LONGJOHNS
3	D	19.04	RAISED CRULLERS
	D	20	DUMPLINGS
	D	20.01	DUMPLINGS (Biscuit Mix)
	D	20.02	OVEN-STEAMED DUMPLINGS
	D	20.03	PARSLEY DUMPLINGS
	D	20.04	STEAMED DUMPLINGS (Steam Cooker Method)
1	D	21	ENGLISH MUFFINS
1	D	21.01	CINNAMON RAISIN ENGLISH MUFFINS
1	D	21.02	ENGLISH MUFFINS (Prep)
2	D	22	FRENCH TOAST
2	D	22.01	CINNAMON FRENCH TOAST
2	D	22.02	FRENCH TOAST (Dehy Egg Mix)
2	D	22.03	THICK SLICE FRENCH TOAST
2	D	22.04	ENGLISH MUFFIN FRENCH TOAST
3	D	23	FRENCH TOAST PUFF
3	D	23.01	CINNAMON FRENCH TOAST PUFF
3	D	23.02	SUGARED FRENCH TOAST PUFF
3	D	24	APPLE FRITTERS
2	D	25	PANCAKES
2	D	25.01	APPLE PANCAKES
2	D	25.02	BLUEBERRY PANCAKES
2	D	25.03	BUCKWHEAT PANCAKES (Pancake Mix)
2	D	25.04	BUTTERMILK PANCAKES (Pancake Mix)
2	D	25.05	PANCAKES (Pancake Mix)
2	D	25.06	PEACH PANCAKES
2	D	25.07	PINEAPPLE PANCAKES
2	D	25.08	WAFFLES
2	D	25.09	WAFFLES, FRZ (Brown & Serve)
2	D	25.10	WAFFLES (Pancake Mix)
3	D	26	HOT CROSS BUNS
2	D	27	KOLACHES
2	D	27.01	KOLACHES (Sweet Dough Mix)
	D	28	BRAN MUFFINS
	D	28.01	RAISIN BRAN MUFFINS
	D	28.02	SHREDDED BRAN MUFFINS
2	D	29	MUFFINS
2	D	29.01	BLUEBERRY MUFFINS
2	D	29.02	FILLED MUFFINS
2	D	29.03	SUGAR-TOPPED MUFFINS
2	D	29.04	APPLE MUFFINS
2	D	29.05	CINNAMON CRUMB TOP MUFFINS
2	D	29.06	CRANBERRY MUFFINS
2	D	29.07	DATE MUFFINS
2	D	29.08	NUT MUFFINS
2	D	29.09	RAISIN MUFFINS
2	D	29.10	BANANA MUFFINS

2	D 30	BANANA BREAD
1	D 31	PLAIN MUFFINS (Muffin Mix)
1	D 31.01	BLUEBERRY MUFFINS (Muffin Mix)
1	D 31.02	FILLED MUFFINS (Muffin Mix)
1	D 31.03	SUGAR-TOPPED MUFFINS (Muffin Mix)
	D 32	HARD ROLLS
	D 33	HOT ROLLS
	D 33.01	HOT ROLLS (Brown & Serve)
	D 33.02	HOT ROLLS (Roll Mix)
	D 34	HOT ROLLS (Short-Time Formula)
	D 34.01	BROWN & SERVE ROLLS (Short-Time Formula)
	D 34.02	WHOLE WHEAT ROLLS (Short-Time Formula)
2	D 35	ONION ROLLS
2	D 35.01	ONION ROLLS (Roll Mix)
	D 36	SWEET DOUGH
	D 36.01	SWEET DOUGH (Sweet Dough Mix)
2	D 37	QUICK COFFEE CAKE (Biscuit Mix)
2	D 37.01	QUICK APPLE COFFEE CAKE (Biscuit Mix)
2	D 37.02	QUICK APRICOT COFFEE CAKE (Biscuit Mix)
2	D 37.03	QUICK CHERRY COFFEE CAKE (Biscuit Mix)
2	D 37.04	QUICK CHERRY-NUT COFFEE CAKE (Biscuit Mix)
2	D 37.05	QUICK CRANBERRY COFFEE CAKE (Biscuit Mix)
2	D 37.06	QUICK FRENCH COFFEE CAKE (Biscuit Mix)
2	D 37.07	QUICK ORANGE-COCONUT COFFEE CAKE (Biscuit Mix)
	D 38	CAKE MUFFINS (Cake Mix)
	D 38.01	BLUEBERRY MUFFINS (Cake Mix)
	D 38.02	FILLED CAKE MUFFINS (Cake Mix)
	D 38.03	SUGAR-TOPPED CAKE MUFFINS (Cake Mix)
2	D 39	BREAD STICKS
2	D 39.01	CARAWAY SEED BREAD STICKS
2	D 39.02	GARLIC BREAD STICKS
2	D 39.03	POPPY SEED BREAD STICKS
2	D 39.04	SESAME SEED BREAD STICKS
1	D 41	CHERRY FILLING (Cornstarch)
1	D 41.01	CHERRY FILLING (Pie Filling, Prep)
1	D 41.02	CHERRY FILLING (Pregelatinized Starch)
	D 42	CINNAMON SUGAR FILLING
	D 42.01	CINNAMON HONEY FILLING
	D 42.02	CINNAMON HONEY FILLING (With Cornmeal)
	D 42.03	CINNAMON SUGAR NUT FILLING
	D 42.04	CINNAMON SUGAR RAISIN FILLING
2	D 43	NUT FILLING
2	D 45	PINEAPPLE SYRUP GLAZE
2	D 45.01	SYRUP GLAZE
3	D 46	VANILLA GLAZE
3	D 46.01	ALMOND GLAZE
3	D 46.02	CHERRY GLAZE
3	D 46.03	LEMON GLAZE
3	D 46.04	MAPLE GLAZE
3	D 46.05	ORANGE GLAZE
3	D 46.06	RUM GLAZE
2	D 48	ORANGE-COCONUT TOPPING

Armed Forces Recipe Service

	D	49	STREUSEL TOPPING
	D	49.01	PECAN TOPPING
	D	49.02	PRALINE TOPPING
	D	49.03	RAISIN TOPPING
	D	50	BROWN SUGAR SYRUP
	D	50.01	HONEY SYRUP
	D	50.02	MAPLE SYRUP
	D	51	FRYING BATTER
2	D	52	TEMPURA BATTER
2	D	52.01	TEMPURA BATTER FOR VEGETABLES
3	D	53	CORNSTARCH WASH
3	D	53.01	BUTTER WASH
2	D	55	EGG WASH
2	D	55.01	EGG WHITE WASH
1	D	60	SUBMARINE ROLLS (Hoagie, Torpedo)
1	D	60.01	SUBMARINE ROLLS (Roll Mix)
1	D	61	DANISH DIAMONDS (Danish Dough, Frz)
1	D	61.01	BEAR CLAWS (Danish Dough, Frz)
2	E	1	HOT BREAKFAST CEREALS (Quick-Cooking Type)
2c	E	2	BUTTERED HOMINY GRITS
2c	E	2.01	FRIED HOMINY GRITS
2c	E	2.02	FRIED CORNMEAL MUSH
2c	E	2.03	FRIED OATMEAL MUSH
2	E	3	BUTTERED HOMINY
2	E	3.01	FRIED HOMINY
2	E	4	BOILED PASTA
2	E	4.01	BUTTERED PASTA
2	E	4.02	LASAGNA NOODLES
2	E	4.03	VERMICELLI
2	E	5	STEAMED RICE
2	E	5.01	LYONNAISE RICE
2	E	5.02	TOSSED GREEN RICE
2	E	5.03	YELLOW RICE W/GREEN PEPPERS, RED PEPPERS, & ONIONS
2	E	5.04	RICE W/PARMESAN CHEESE
1	E	7	FRIED RICE (Oven Method)
1	E	7.01	FRIED RICE (Griddle Method)
1	E	7.02	FILIPINO RICE
1	E	7.03	PORK FRIED RICE
1	E	7.04	SHRIMP FRIED RICE
	E	8	RICE PILAF
	E	8.01	ORANGE RICE
	E	8.02	RICE PILAF (Soup, Dehy, Onion)
1	E	9	SPANISH RICE
	E	10	RED BEANS W/RICE
	E	10.01	HOPPING JOHN (Black-eye Peas w/Rice)
2	E	11	MEXICAN RICE
	E	12	NOODLES JEFFERSON
2	E	13	NOODLES W/PESTO SCE
1	E	14	STEAMED PASTA
1	E	14.01	STEAMED LASAGNA NOODLES
1	E	15	STEAMED RICE (Steam Cooker Method)
3	F	1	BAKED MACARONI & CHEESE

3	F	1.01	BAKED MACARONI & CHEESE (Soup, Cond, Cream of Mushroom)
3	F	1.02	BAKED EGG NOODLES & CHEESE
3	F	2	WELSH RAREBIT
3	F	2.01	TOMATO RAREBIT
3	F	3	SCALLOPED NOODLES W/CHEESE, TOMATOES, & BACON
3	F	3.01	SCALLOPED MACARONI W/CHEESE, TOMATOES, & BACON
1	F	4	COOKED EGGS
1	F	4.01	COLD WATER METHOD FOR COOKED EGGS
1	F	4.02	STEAMER METHOD FOR COOKING EGGS
3	F	5	DEVILED EGGS
2	F	6	EGG FOO YOUNG
1	F	7	GRIDDLE FRIED EGGS (Cooked to Order)
1	F	7.01	OVEN-FRIED EGGS
3	F	8	PLAIN OMELET
3	F	8.01	BACON OMELET
3	F	8.02	BAUERNFRUESTUECK (Farmer's Breakfast)
3	F	8.03	CHEESE OMELET
3	F	8.04	GREEN PEPPER OMELET
3	F	8.05	HAM OMELET
3	F	8.06	HAM & CHEESE OMELET
3	F	8.07	INDIVIDUAL OMELET
3	F	8.08	MUSHROOM OMELET
3	F	8.09	ONION OMELET
3	F	8.10	WESTERN OMELET
3	F	8.11	TOMATO OMELET
3	F	8.12	SPANISH OMELET
3	F	9	POACHED EGGS
1	F	10	SCRAMBLED EGGS
1	F	10.01	SCRAMBLED EGGS & CHEESE
1	F	10.02	SCRAMBLED EGGS & HAM
1	F	10.03	SCRAMBLED EGGS (Dehy Egg Mix)
1	F	10.04	SCRAMBLED EGGS (Single Serving)
1	F	11	ONION & MUSHROOM QUICHE
3	F	12	CREAMED EGGS
3	F	12.01	EGGS AU GRATIN (Scotch Woodcock)
	G	1	ANGEL FOOD CAKE (Cake Mix)
	G	3	APPLESAUCE CAKE (Yellow Cake Mix)
	G	4	PUMPKIN CAKE
	G	6	BANANA CAKE (Yellow Cake Mix)
	G	6.01	BANANA CAKE (Banana Cake Mix)
1	G	8	FLORIDA LEMON CAKE
1	G	9	CHOCOLATE CAKE
1	G	9.01	CHOCOLATE CAKE (Cake Mixes)
1	G	9.02	GERMAN CHOCOLATE CAKE (Cake Mix)
1	G	10	YELLOW CAKE (Yellow Cake Mix)
1	G	10.01	ALMOND FLAVORED CAKE (Yellow Cake Mix)
1	G	10.02	BLACK WALNUT CAKE (Yellow Cake Mix)
1	G	10.03	LEMON CAKE (Yellow Cake Mix)
1	G	10.04	MAPLE NUT CAKE (Yellow Cake Mix)
1	G	10.05	MARBLE CAKE (Yellow Cake Mix)
1	G	10.06	ORANGE CAKE (Yellow Cake Mix)
	G	11	EASY CHOCOLATE CAKE

Armed Forces Recipe Service

	G 12	DEVIL'S FOOD CAKE
	G 12.01	DEVIL'S FOOD CAKE (Cake Mix)
1	G 13	CARROT CAKE (Cake Mix)
	G 16	FRUIT SHORTCAKE (Biscuit Base)
	G 16.01	FRUIT SHORTCAKE (Biscuit Mix)
	G 16.02	FRUIT SHORTCAKE (Cake Base)
	G 16.03	FRUIT SHORTCAKE (Cake Mix)
	G 17	GINGERBREAD
	G 17.01	GINGERBREAD (Gingerbread Cake Mix)
3	G 18	JELLY ROLL
3	G 20	PEANUT BUTTER CRUNCH CAKE
3	G 20.01	PEANUT BUTTER CAKE
3	G 21	POUND CAKE
3	G 21.01	ALMOND POUND CAKE (Pound Cake Mix)
3	G 21.02	VELVET POUND CAKE (Yellow Cake Mix)
3	G 21.03	LEMON POUND CAKE (Pound Cake Mix)
	G 24	RAISIN NUT CAKE
3	G 25	SPICE CAKE
3	G 25.01	SPICE CAKE (Yellow Cake Mix)
1	G 27	CHOCOLATE CHIP CAKE
1	G 27.01	CHOCOLATE CHIP CAKE (Yellow Cake Mix)
	G 28	STRAWBERRY CAKE (White Cake Mix)
2	G 29	PINEAPPLE UPSIDE DOWN CAKE (Sliced Pineapple)
2	G 29.01	PINEAPPLE UPSIDE DOWN CAKE (Cake Mix)
2	G 29.02	PINEAPPLE UPSIDE DOWN CAKE (Crushed Pineapple)
2	G 29.03	FRUIT COCKTAIL UPSIDE DOWN CAKE
2	G 29.04	PINEAPPLE UPSIDE DOWN CAKE (Crushed Pineapple & Cake Mix)
2	G 29.05	FRUIT COCKTAIL UPSIDE DOWN CAKE (Cake Mix)
1	G 30	WHITE CAKE
1	G 30.01	LEMON-FILLED CAKE
1	G 30.02	NUT CAKE (White Cake Mix)
1	G 30.03	PINEAPPLE-FILLED CAKE
1	G 30.04	WHITE CAKE (White Cake Mix)
1	G 30.05	LEMON-FILLED CAKE (White Cake Mix)
1	G 30.06	PINEAPPLE-FILLED CAKE (White Cake Mix)
1	G 30.07	RASPBERRY-FILLED CAKE (White Cake Mix)
1	G 30.08	STRAWBERRY-FILLED CAKE (White Cake Mix)
1	G 30.09	RASPBERRY-FILLED CAKE
1	G 30.10	STRAWBERRY-FILLED CAKE
3	G 32	YELLOW CAKE
3	G 32.01	BANANA-FILLED LAYER CAKE
3	G 32.02	BOSTON CREAM PIE
3	G 32.03	CHOCOLATE CREAM CAKE
3	G 32.04	COCONUT CAKE
3	G 32.05	COTTAGE PUDDING
3	G 32.06	DUTCH APPLE CAKE
3	G 32.07	FILLED CAKE (Washington Pie)
3	G 32.08	MARBLE CAKE
1	G 33	COCONUT PECAN FROSTING
1	G 34	CREAM CHEESE FROSTING
2	G 35	CHEESE CAKE
2	G 35.01	CHEESE CAKE (Cheese Cake Mix)

2	G	35.02	CHEESE CAKE PIE (Cheese Cake Mix)
2	G	35.03	CHEESE CAKE W/BLUEBERRY TOPPING
2	G	35.04	CHEESE CAKE W/CHERRY TOPPING
2	G	35.05	CHEESE CAKE W/PEACH TOPPING
2	G	35.06	CHEESE CAKE W/SOUR CREAM TOPPING
2	G	35.07	CHEESE CAKE W/STRAWBERRY TOPPING
2	G	35.08	CHEESE CAKE W/BLUEBERRY TOPPING (Cheese Cake Mix)
2	G	35.09	CHEESE CAKE W/CHERRY TOPPING (Cheese Cake Mix)
2	G	35.10	CHEESE CAKE W/PEACH TOPPING (Cheese Cake Mix)
	C	37	PINEAPPLE FILLING (Pregelatinized Starch)
	G	37.01	PINEAPPLE FILLING (Cornstarch)
3	G	39	BUTTER CREAM FROSTING
3	G	39.01	BANANA BUTTER CREAM FROSTING
3	G	39.02	CHOCOLATE BUTTER CREAM FROSTING
3	G	39.03	COCONUT BUTTER CREAM FROSTING
3	G	39.04	LEMON BUTTER CREAM FROSTING
3	G	39.05	MAPLE BUTTER CREAM FROSTING
3	G	39.06	MOCHA BUTTER CREAM FROSTING
3	G	39.07	ORANGE BUTTER CREAM FROSTING
	G	40	BROWN SUGAR FROSTING
1	G	41	CARAMEL FROSTING
2	G	42	CHOCOLATE GLAZE FROSTING
2	G	44	CHOCOLATE FUDGE FROSTING
	G	46	FRENCH CREAM FROSTING
	G	47	LIGHT BUTTER CREAM FROSTING
	G	49	PEANUT BUTTER CREAM FROSTING
1	G	50	CHOCOLATE FROSTING (Icing Mix, Chocolate, Powdered)
1	G	50.01	CHOCOLATE CHIP FROSTING (Icing Mix, Chocolate, Powdered)
1	G	50.02	CHOCOLATE COCONUT FROSTING (Icing Mix, Chocolate, Powdered)
1	G	50.03	CHOCOLATE MARSHMALLOW FROSTING (Icing Mix, Chocolate, Powdered)
1	G	50.04	MOCHA CREAM FROSTING (Icing Mix, Chocolate, Powdered)
	G	51	DECORATOR'S FROSTING
3	G	52	VANILLA FROSTING (Icing Mix, Vanilla, Powdered)
3	G	52.01	MAPLE WALNUT FROSTING (Icing Mix, Vanilla, Powdered)
3	G	52.02	ORANGE FROSTING (Icing Mix, Vanilla, Powdered)
3	G	52.03	PECAN FROSTING (Icing Mix, Vanilla, Powdered)
	G	53	CHOCOLATE CHIP FUDGE FROSTING
	G	54	CHOCOLATE MACAROON CAKE (Cake Mixes)
1	H	1	APPLE CAKE BROWNIES
1	H	11	APPLE CAKE BROWNIES (Gingerbread Cake Mix)
1	H	12	APPLE CAKE BROWNIES (Yellow Cake Mix)
2	H	2	BROWNIES
2	H	2.01	BROWNIES (Chocolate Brownie Mix)
2	H	2.02	CAKE BROWNIES
2	H	2.03	PEANUT BUTTER BROWNIES
	H	3	BUTTERSCOTCH BROWNIES
	H	3.01	BUTTERSCOTCH BROWNIES (Butterscotch Brownie Mix)
	H	4	CHEWY NUT BARS
	H	4.01	CONGO BARS
	H	5	SHORTBREAD COOKIES
	H	6	CRISP TOFFEE BARS

Armed Forces Recipe Service

	H	7	FRUIT BARS
	H	7.01	GINGER FRUIT BARS (Oatmeal Cookie Mix & Gingerbread Cake Mix)
3	H	8	GINGERBREAD COOKIES
1	H	9	OATMEAL COOKIES (Oatmeal Cookie Mix)
1	H	9.01	OATMEAL APPLESAUCE COOKIES (Oatmeal Cookie Mix)
1	H	9.02	OATMEAL CHOCOLATE CHIP COOKIES (Oatmeal Cookie Mix)
1	H	9.03	OATMEAL DATE COOKIES (Oatmeal Cookie Mix)
1	H	9.04	OATMEAL RAISIN COOKIES (Oatmeal Cookie Mix)
1	H	9.05	SPICED OATMEAL NUT COOKIES (Oatmeal Cookie Mix)
1	H	10	CRISP CHOCOLATE COOKIES
1	H	10.01	CHOCOLATE COOKIES (Chocolate Cookie Mix)
1	H	10.02	DOUBLE CHOCOLATE CHIP BARS (Chocolate Cookie Mix)
1	H	10.03	DOUBLE CHOCOLATE CHIP COOKIES (Chocolate Cookie Mix)
	H	11	BANANA DROP COOKIES
1	H	12	CHOCOLATE DROP COOKIES
1	H	12.01	CHOCOLATE DROP COOKIES (Chocolate Brownie Mix)
3	H	13	SUGAR COOKIES
3	H	13.01	SUGAR COOKIES (Sugar Cookie Mix)
3	H	13.02	SNICKERDOODLE COOKIES
3	H	13.03	SNICKERDOODLE COOKIES (Sugar Cookie Mix)
	H	14	COCONUT RAISIN DROP COOKIES
	H	15	CRISP DROP COOKIES
	H	15.01	BROWN SUGAR DROP COOKIES
	H	15.02	COCONUT DROP COOKIES
	H	15.03	LEMON DROP COOKIES
	H	15.04	RAISIN DROP COOKIES
	H	16	COCONUT CEREAL COOKIES
	H	17	HERMITS
	H	18	FRUIT NUT BARS
1	H	20	CHOCOLATE CHIP COOKIES
1	H	20.01	CHOCOLATE CHIP COOKIES (Sugar Cookie Mix)
1	H	20.02	CHOCOLATE CHIP NUT COOKIES
1	H	20.03	CHOCOLATE CHIP BARS (Sugar Cookie Mix)
	H	21	LEMON COOKIES
	H	21.01	ALMOND COOKIES
	H	21.02	ORANGE COOKIES
	H	21.03	VANILLA COOKIES
3	H	22	GINGER MOLASSES COOKIES
3	H	22.02	GINGER MOLASSES BARS
1	H	23	OATMEAL COOKIES
1	H	23.01	OATMEAL CHOCOLATE CHIP COOKIES
1	H	23.02	OATMEAL NUT COOKIES
1	H	24	PEANUT BUTTER COOKIES
1	H	24.01	PEANUT BUTTER COOKIES (Sugar Cookie Mix)
1	H	24.02	PEANUT BUTTER BARS (Sugar Cookie Mix)
3	I	1	PIE CRUST
3	I	1.01	PIE CRUST (Dough Rolling Method)
3	I	1.02	PIE CRUST (Manual Method)
3	I	2	GRAHAM CRACKER CRUST
3	I	2.01	GRAHAM CRACKER CRUST (Preformed Crust)
3	I	3	MINCEMEAT PIE
3	I	4	EGG & MILK WASH

3	I	4.01	EGG & WATER WASH
3	I	4.02	MILK & WATER WASH
3	I	5	MERINGUE
3	I	5.01	MERINGUE (Dehy)
3	I	6	VANILLA CREAM PIE
3	I	6.01	BANANA CREAM PIE
3	I	6.02	COCONUT CREAM PIE
3	I	6.03	STRAWBERRY GLAZED CREAM PIE
3	I	6.04	PINEAPPLE CREAM PIE
3	I	7	VANILLA CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	7.01	BANANA CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	7.02	COCONUT CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	7.03	PINEAPPLE CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	7.04	STRAWBERRY GLAZED CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	8	APPLE PIE (Cnd Apples-Pregelatinized Starch)
3	I	8.01	DUTCH APPLE PIE (Cnd Apples-Pregelatinized Starch)
3	I	8.02	FRENCH APPLE PIE (Cnd Apples-Pregelatinized Starch)
3	I	8.03	APPLE PIE (Dehy Apples)
3	I	9	APPLE PIE (Cnd Apples-Cornstarch)
3	I	9.01	APPLE PIE (Pie Filling, Prep)
3	I	9.02	APPLE PIE (Dehy Apples-Cornstarch)
3	I	9.03	APPLE PIE (Dehy Apple Pie Filling Mix)
3	I	9.04	APPLE PIE (Dehy Apple Pie Filling, Comp)
3	I	9.05	DUTCH APPLE PIE (Pie Filling, Prep)
3	I	10	APPLE COBBLER
3	I	10.01	BLACKBERRY COBBLER
3	I	10.02	BLUEBERRY COBBLER
3	I	10.03	CHERRY COBBLER
3	I	10.04	STREUSEL-TOPPED APPLE COBBLER
3	I	10.05	PEACH COBBLER
2	I	11	CHOCOLATE MOUSSE PIE
3	I	12	SWEET POTATO PIE
3	I	13	PUMPKIN PIE
3	I	14	PINEAPPLE PIE (Cnd Pineapple-Cornstarch)
3	I	15	BERRY PIE (Frz Berries-Cornstarch)
3	I	15.01	BLUEBERRY PIE (Blueberries, IQF-Cornstarch)
3	I	16	BLUEBERRY PIE (Cnd Blueberries-Pregelatinized Starch)
3	I	16.01	BLACKBERRY PIE (Cnd Blackberries-Pregelatinized Starch)
3	I	17	BLUEBERRY PIE (Cnd Blueberries-Cornstarch)
3	I	17.01	BLACKBERRY PIE (Cnd Blackberries-Cornstarch)
3	I	17.02	BLUEBERRY PIE (Pie Filling, Prep)
3	I	18	PINEAPPLE PIE (Cnd Pineapple-Pregelatinized Starch)
3	I	19	BUTTERSCOTCH CREAM PIE
3	I	19.01	BUTTERSCOTCH CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	20	PEACH PIE (Frz Peaches-Cornstarch)
3	I	21	CHERRY PIE (Cnd Cherries-Pregelatinized Starch)
3	I	22	CHERRY PIE (Cnd Cherries-Cornstarch)
3	I	22.01	CHERRY PIE (Pie Filling, Prep)
3	I	23	PEACH PIE (Frz Peaches-Pregelatinized Starch)
3	I	24	PEACH PIE (Cnd Peaches-Cornstarch)
3	I	24.01	PEACH PIE (Pie Filling, Prep)
3	I	25	PEACH PIE (Cnd Peaches-Pregelatinized Starch)

Armed Forces Recipe Service

3	I	26	CREAMY COCONUT PIE
3	I	26.01	CREAMY BANANA COCONUT PIE
3	I	26	AMBROSIA PIE
3	I	27	CHERRY CRUMBLE PIE
3	I	28	CHOCOLATE CREAM PIE
3	I	28.01	CHOCOLATE CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	28.02	CHOCOLATE COCONUT CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	28.03	CHOCOLATE NUT CREAM PIE (Dessert Powder, Pudding, Instnt)
3	I	30	FRIED APPLE PIE (Dehy Apples)
3	I	30.01	FRIED APPLE PIE (Pie Filling, Prep)
3	I	30.02	FRIED CHERRY PIE (Pie Filling, Prep)
3	I	30.03	FRIED PEACH PIE (Pie Filling, Prep)
3	I	30.04	FRIED BLUEBERRY PIE (Pie Filling, Prep)
3	I	30.05	FRIED LEMON PIE (Pie Filling, Prep)
3	I	32	LEMON CHIFFON PIE
3	I	32.01	PINEAPPLE CHIFFON PIE
3	I	32.02	STRAWBERRY CHIFFON PIE
3	I	33	LEMON MERINGUE PIE
3	I	33.01	LEMON MERINGUE PIE (Pie Filling, Prep)
3	I	33.02	LEMON MERINGUE PIE (Pie Filling, Prep Mix, Lemon Flavored, RTC)
	I	40	PECAN PIE
	I	40.01	WALNUT PIE
1	I	48	CHOCOLATE & VANILLA PIE (Dessert Powder, Pudding, Instnt)
1	J	1	APPLE CRISP
1	J	1.01	APPLE CRISP (Dehy Apples)
1	J	1.02	APPLE CRISP (Oatmeal Cookie Mix)
1	J	1.03	CHEESE APPLE CRISP
1	J	1.04	CRUNCHY APPLE CRISP
1	J	1.05	APPLE CRISP (Pie Filling, Prep)
	J	2	APPLESAUCE CRISP
	J	2.01	APPLESAUCE CRISP (Oatmeal Cookie Mix)
3	J	3	BAKED APPLES
3	J	3.01	BAKED APPLES W/RAISIN NUT FILLING
3	J	3.02	BAKED APPLES W/RAISIN COCONUT FILLING
	J	4	VANILLA SOFT SERVE ICE CREAM (Mix, Liquid, Fresh, Vanilla)
3	J	4.01	STRAWBERRY SOFT SERVE ICE CREAM (Mix, Liquid, Fresh, Vanilla)
3	J	4.02	VANILLA MILK SHAKE (Mix, Liquid, Fresh, Vanilla)
3	J	4.03	CHOCOLATE MILK SHAKE (Mix, Liquid, Fresh, Chocolate)
3	J	4.04	CHOCOLATE SOFT SERVE ICE CREAM (Mix, Liquid, Fresh, Chocolate)
3	J	4.05	FRUIT FLAVORED SOFT SERVE ICE CREAM (Mix, Liquid, Fresh)
3	J	5	FLUFFY FRUIT CUP
3	J	5.01	YOGURT FRUIT CUP
3	J	5.02	SOUR CREAM FRUIT CUP
3	J	6	FRUIT CUP
3	J	6.01	AMBROSIA
3	J	6.02	BANANA FRUIT CUP
3	J	6.03	MELON (Cantaloupe, Honeydew or Watermelon) FRUIT CUP
3	J	6.04	STRAWBERRY FRUIT CUP
3	J	6.05	FRUIT COCKTAIL FRUIT CUP
3	J	6.06	SPICED FRUIT CUP
3	J	7	FRUIT GELATIN

Armed Forces Recipe Service

3	J	7.01	BANANA GELATIN
3	J	7.02	FRUIT FLAVORED GELATIN
3	J	7.03	FRUIT GELATIN (Crushed Ice Method)
3	J	7.04	STRAWBERRY GEALTIN
3	J	7.05	PEACH GELATIN
3	J	8	PEACH CRISP
3	J	8.01	CHERRY CRISP (Pie Filling, Prep)
3	J	8.02	CHERRY CRISP
3	J	8.03	PEACH CRISP (Oatmeal Cookie Mix)
3	J	8.04	PEACH CRISP (Pie Filling, Prep)
3	J	8.05	PEACH CRISP (Pie Filling, Prep & Oatmeal Cookie Mix)
3	J	8.06	BLUEBERRY CRISP (Pie Filling, Prep)
2	J	9	STEWED PRUNES
2	J	9.01	STEWED PRUNES (Cnd Prunes)
3	J	11	BANANA SPLIT
3	J	12	VANILLA ICE CREAM
3	J	12.01	CHOCOLATE ICE CREAM
3	J	12.02	STRAWBERRY ICE CREAM
1	J	13	TAPIOCA PUDDING
	J	14	BAKED CUSTARD
	J	14.01	BAKED CARAMEL CUSTARD
	J	14.02	BAKED CHOCOLATE CUSTARD
	J	14.03	BAKED COCONUT CUSTARD
	J	14.04	BREAD CUSTARD PUDDING
3	J	15	BAKED RICE PUDDING
3	J	15.01	BAKED COCONUT RICE PUDDING
	J	16	BREAD PUDDING
	J	16.01	CHOCOLATE CHIP BREAD PUDDING
	J	16.02	COCONUT BREAD PUDDING
	J	18	CHERRY CAKE PUDDING
	J	18.01	CHERRY CAKE PUDDING (Cake Mix)
	J	19	CHOCOLATE CAKE PUDDING
	J	19.01	CHOCOLATE CAKE PUDDING (Cake Mix)
	J	20	FRUIT COCKTAIL CAKE PUDDING
	J	20.01	FRUIT COCKTAIL CAKE PUDDING (Cake Mix)
3	J	21	VANILLA CREAM PUDDING (Dessert Powder, Pudding, Instnt)
3	J	21.01	BANANA CREAM PUDDING (Dessert Powder, Pudding, Instnt)
3	J	21.02	COCONUT CREAM PUDDING (Dessert Powder, Pudding, Instnt)
3	J	21.03	PINEAPPLE CREAM PUDDING (Dessert Powder, Pudding, Instnt)
3	J	21.04	BUTTERSCOTCH CREAM PUDDING (Dessert Powder, Pudding, Instnt)
3	J	21.05	CHOCOLATE CREAM PUDDING (Dessert Powder, Pudding, Instnt)
3	J	21.06	VANILLA CREAM PUDDING (Cnd, Prep)
3	J	21.07	CHOCOLATE CREAM PUDDING (Cnd, Prep)
	J	22	CREAMY RICE PUDDING
3	J	24	VANILLA SOFT SERVE ICE CREAM (Ice Milk-Shake Mix, Dehy)
3	J	24.01	CHOCOLATE SOFT SERVE ICE CREAM (Ice Milk-Shake Mix, Dehy)
3	J	24.02	CHOCOLATE MILK SHAKE (Ice Milk-Shake Mix, Dehy)
3	J	24.03	STRAWBERRY SOFT SERVE ICE CREAM (Ice Milk-Shake Mix, Dehy)
3	J	24.04	VANILLA MILK SHAKE (Ice Milk-Shake Mix, Dehy)
3	J	24.05	CHERRY MILK SHAKE (Ice Milk-Shake Mix, Dehy)
3	J	24.06	ORANGE MILK SHAKE (Ice Milk-Shake Mix, Dehy)
3	J	24.07	STRAWBERRY MILK SHAKE (Ice Milk-Shake Mix, Dehy)

3	J	25	VANILLA CREAM PUDDING
3	J	25.01	CHOCOLATE CREAM PUDDING
2	J	26	CREAM PUFFS
2	J	26.01	ECLAIRS
2	J	27	PINEAPPLE CRUNCH
2	J	27.01	APPLE CRUNCH (Pie Filling, Prep)
2	J	27.02	BLUEBERRY CRUNCH (Pie Filling, Prep)
2	J	27.03	CHERRY CRUNCH (Pie Filling, Prep)
2	J	27.04	CHERRY CRUNCH (Cnd Red Tart [Sour] Cherries)
2	J	27.05	PEACH CRUNCH (Pie Filling, Prep)
2	J	27.06	CHERRY CRUNCH (Red Sour Cherries, Frz)
2	J	27.07	PEACH CRUNCH (Peaches, Frz)
1	J	32	VANILLA SOFT SERVE YOGURT
1	J	32.01	FRUIT FLAVORED SOFT SERVE YOGURT (Fresh Liquid Yogurt Mix)
2	K	1	PEACH SCE
2	K	1.01	APRICOT SCE
2	K	1.02	PEACH SCE (Pie Filling, Prep)
3	K	2	WHIPPED TOPPING
3	K	2.01	WHIPPED TOPPING (Topping, Dessert & Bakery Products, Frz)
	K	3	CARAMEL SCE
2	K	4	CHERRY SCE
2	K	5	CHOCOLATE SCE
2	K	5.01	CHOCOLATE COCONUT SCE
2	K	5.02	CHOCOLATE MARSHMALLOW SCE
2	K	5.03	CHOCOLATE NUT SCE
2	K	5.04	CHOCOLATE PEANUT BUTTER SCE
2	K	5.05	CHOCOLATE RUM SCE
2	K	5.06	CHOCOLATE BLACK WALNUT SCE
2	K	5.07	CHOCOLATE MINT SCE
2	K	5.08	CHOCOLATE ORANGE SCE
2	K	5.09	CHOCOLATE ALMOND SCE
1	K	9	LEMON SCE
1	K	10	ORANGE SCE
	K	11	PINEAPPLE SCE
3	K	12	VANILLA SCE
	K	14	CHERRY JUBILEE SCE
	K	15	WHIPPED CREAM
1	K	18	RUM SCE
3	L	1	OVEN FRIED BACON SLICES (Preckd Bacon)
3	L	1.01	GRILLED BACON (Preckd Bacon)
1	L	2	OVEN FRIED BACON
1	L	2.01	OVEN FRIED CANADIAN BACON
1	L	2.02	GRILLED BACON
1	L	2.03	GRILLED CANADIAN BACON
3	L	3	CHICKEN ENCHILADAS
3	L	4	ROAST RIB OF BEEF (Beef, Rib, Bone-in)
3	L	4.01	STEAMSHIP ROUND OF BEEF (Beef, Round, Bone-In)
3	L	4.02	STEAMSHIP ROUND OF BEEF (Beef, Round, Boneless)
3	L	4.03	ROAST RIB OF BEEF (Beef, Boneless, Ribeye Roll)
3	L	5	ROAST BEEF
3	L	5.01	ROAST BEEF (Oven-Cooked)
3	L	6	SUKIYAKI

3	L	7	GRILLED STEAK
3	L	7.01	GRILLED TENDERLOIN STEAK
1	L	8	TERIYAKI STEAK
1	L	10	BEEF POT ROAST
1	L	10.01	GINGER POT ROAST
1	L	10.02	YANKEE POT ROAST
3	L	11	SIMMERED BEEF
2	L	12	CHICKEN FRIED STEAKS
2	L	12.01	CHICKEN FRIED STEAKS (Dehy Beef Steak)
3	L	13	PEPPER STEAK
3	L	13.01	ORIENTAL STEAK
	L	14	SPANISH STEAK
	L	14.01	SPANISH STEAK STRIPS
	L	15	STEAK SMOTHERED W/ONIONS
	L	15.01	STEAK STRIPS SMOTHERED W/ONIONS
3	L	16	SWISS STEAK W/TOMATO SCE
3	L	16.01	SWISS STEAK W/BROWN GRAVY
3	L	16.02	SWISS STEAK W/TOMATO SCE (Sce Mix, Tomato Basic)
3	L	16.03	SWISS STEAK W/TOMATO SCE (Soup, Cond, Tomato)
3	L	16.04	SWISS STEAK W/MUSHROOM GRAVY
1	L	17	BRAISED BEEF & NOODLES
1	L	17.01	BRAISED BEEF CUBES
3	L	18	BARBECUED BEEF CUBES
3	L	18.01	BARBECUED BEEF CUBES (Cnd Beef Chunks w/Natural Jces)
1	L	19	STUFFED FLOUNDER CREOLE
3	L	20	BEEF & CORN PIE
	L	21	BEEF POT PIE
	L	21.01	BEEF POT PIE W/PIE CRUST TOPPING
	L	21.02	BEEF POT PIE W/THIN BATTER TOPPING
1	L	22	BEEF STEW
1	L	22.01	BEEF STEW (Cnd)
2	L	23	EL RANCHO STEW
1	L	24	STUFFED CABBAGE ROLLS
1	L	24.01	STUFFED CABBAGE ROLLS (Soup, Cond, Tomato)
3	L	25	LASAGNA
3	L	25.01	LASAGNA (Sce Mix, Tomato Basic)
3	L	25.02	LASAGNA (Frz)
3	L	25.03	LASAGNA (Pizza Sce, Cnd)
1	L	26	SYRIAN BEEF STEW
3	L	27	BEEF BALLS STROGANOFF
2	L	28	CHILI CON CARNE
2c	L	28.01	CHILI CON CARNE (Sce Mix, Tomato Basic)
2c	L	28.02	CHILI MACARONI
2c	L	28.03	CHILI MACARONI (Sce Mix, Tomato Basic)
1	L	29	BEEF PORCUPINES
1	L	29.01	BEEF PORCUPINES (Soup, Cond, Tomato)
3	L	30	CREAMED GROUND BEEF
3	L	31	BEEF TURNOVERS
2	L	33	ROAST BEEF HASH
2	L	33.01	HASH, ROAST BEEF (Cnd)
2	L	33.02	HASH, ROAST BEEF (Cnd Beef Chunks w/Natural Jces)
2	L	33.03	ROAST BEEF HASH (Breakfast Portion)

Armed Forces Recipe Service

1	L 35	MEAT LOAF
1	L 35.01	PORK LOAF
1	L 35.02	TOMATO MEAT LOAF (Soup, Cond, Tomato)
1	L 35.03	VEAL LOAF
3	L 36	MINCED BEEF
2	L 37	SALISBURY STEAK
2	L 37.01	GRILLED SALISBURY STEAK
2	L 38	SPAGHETTI SCE
2	L 38.01	SPAGHETTI SCE (Sce Mix, Tomato Basic)
2	L 39	SPAGHETTI W/MEATBALLS
2	L 39.01	MEATBALLS & SCE (Sce Mix, Tomato Basic)
2	L 39.02	SPAGHETTI W/MEATBALLS (Sce Mix, Tomato Basic)
3	L 40	STUFFED GREEN PEPPERS
3	L 40.01	STUFFED GREEN PEPPERS (Frz Stuffed Peppers)
3	L 41	SWEDISH MEATBALLS
3	L 41.01	SWEDISH MEATBALLS (Beef & Veal)
3	L 42	CHILI CONQUISTADOR
1	L 44	TURKEY CURRY (Turkey, Boneless, Frz, Cooked)
1	L 44.01	TURKEY CURRY (Turkey, Boneless, Frz, Raw)
3	L 45	STUFFED BEEF ROLLS
3	L 45.01	BEEF BROGUL
3	L 47	BEEF PIE W/BISCUIT TOPPING (Cnd Beef Chunks w/Natural Jces)
2	L 48	BEEF & TOMATO GRAVY (Cnd Beef Chunks w/Natural Jces)
	L 49	BAKED SPANISH BEEF PATTIES (Cnd Hamburgers)
	L 49.01	BAKED SPANISH BEEF PATTIES (Dehy Beef Patties)
	L 50	BEEF PATTIES JARDINIERE (Cnd Hamburgers)
	L 50.01	BEEF PATTIES JARDINIERE (Dehy Beef Patties)
1	L 52	CREAMED CHIPPED BEEF
1	L 53	BEEF STROGANOFF
1	L 53.01	BEEF STROGANOFF (Soup, Cond, Cream of Mushroom)
1	L 53.02	HAMBURGER STROGANOFF
	L 55	BEEF CORDON BLEU
3	L 57	TAMALE PIE
3	L 57.01	HOT TAMALES W/CHILI GRAVY
3	L 57.02	TAMALE PIZZA
3	L 57.03	TAMALE PIE (Veal, Ground)
3	L 57.04	TAMALE PIE (Cnd Hamburgers)
	L 58	CHILI & MACARONI (Chili Con Carne, Cnd)
2	L 59	CHILI CON CARNE W/BEANS (Chili Con Carne, Cnd)
2	L 59.01	CHILI CON CARNE W/BEANS (Dehy)
3	L 60	HAMBURGER PARMESAN
3	L 61	TEXAS HASH
1	L 62	YAKISOBA (Beef & Spaghetti)
1	L 62.01	HAMBURGER YAKISOBA (Beef Pattie Mix or Beef, Ground, Bulk)
1	L 63	ENCHILADAS
1	L 63.01	MUSHROOM & OLIVE ENCHILADAS
1	L 63.02	ENCHILADAS (Frz Enchiladas)
	L 64	CREOLE MACARONI (American Chop Suey)
2	L 65	HUNGARIAN GOULASH
1	L 66	SAUERBRATEN
1	L 66.01	SAUERBRATEN (Steam-Jacketed Kettle Method)
3	L 67	GLAZED HAM LOAF

2	L	68	SCALLOPED HAM & NOODLES
2	L	68.01	SCALLOPED HAM & MACARONI
2	L	69	BAKED HAM
2	L	69.01	GRILLED HAM STEAK
2	L	70	BARBECUED HAM STEAK
2	L	70.01	BARBECUED HAM STEAK (Cnd Ham)
2	L	71	BAKED CND HAM
2	L	71.01	BAKED HAM STEAK
2	L	71.02	GRILLED HAM STEAK (Cnd Ham)
2	L	71.03	GRILLED HAM SLICE
2	L	72	BAKED HAM, MACARONI & TOMATOES (Cnd Ham)
2	L	72.01	BAKED LUNCHEON MEAT, MACARONI, & CHEESE
2	L	72.02	BAKED HAM, MACARONI & TOMATOES (Cnd Ham Chunks)
	L	73	SCALLOPED HAM & POTATOES (Cnd Ham Chunks)
2	L	74	SESAME CHICKEN
2	L	75	CHILIBURGERS (Cnd Hamburgers wo/Gravy)
1	L	76	BEEF MANICOTTI (Cannelloni)
1	L	76.01	CHEESE MANICOTTI
3	L	77	SAVORY ROAST LAMB
2	L	78	CHICKEN ADOBO
2	L	79	SWEET & SOUR PORK CHOPS
2	L	79.01	SWEET & SOUR CHICKEN
1	L	80	PORK CHOP SUEY
1	L	80.01	SHRIMP CHOP SUEY
1	L	81	ROAST PORK
3	L	82	SWEET & SOUR PORK
1	L	83	BAKED PORK CHOPS
1	L	83.01	CREOLE PORK CHOPS
1	L	83.02	PORK CHOPS W/APPLE RINGS
1	L	83.03	CREOLE PORK STEAKS (Frz Breaded Pork Steaks)
1	L	84	BAKED STUFFED PORK CHOPS
1	L	85	BRAISED PORK CHOPS
1	L	85.01	GRILLED PORK CHOPS
1	L	85.02	PORK CHOPS W/MUSHROOM GRAVY
1	L	86	BREADED PORK CHOPS
1	L	86.01	BREADED PORK CHOPS (Dehy)
1	L	86.02	BREADED PORK STEAKS (Frz Breaded Pork Steaks)
1	L	86.03	PORK SCHNITZEL (Frz Breaded Pork Steaks)
1	L	87	PORK CHOPS MEXICANA
1	L	88	GRILLED POLISH SAUSAGE
1	L	88.01	BAKED ITALIAN SAUSAGES (Hot or Sweet)
1	L	88.02	GRILLED FRANKFURTERS
1	L	88.03	GRILLED BOCKWURST
1	L	88.04	GRILLED BRATWURST
3	L	89	GRILLED SAUSAGE PATTIES
3	L	89.01	BAKED SAUSAGE PATTIES
3	L	89.02	GRILLED SAUSAGE PATTIES (Preformed)
3	L	89.03	BAKED SAUSAGE PATTIES (Preformed)
	L	90	SIMMERED CHITTERLINGS
	L	90.01	DEEP FAT FRIED CHITTERLINGS
	L	90.02	DEEP FAT FRIED CHITTERLINGS, PRECKD
	L	90.03	SIMMERED CHITTERLINGS, PRECKD

	L 91	GRILLED SAUSAGE LINKS
	L 91.01	BAKED SAUSAGE LINKS
1	L 92	BARBECUED SPARERIBS
1	L 92.01	BARBECUED SPARERIBS (Prep Barbecue Sce)
	L 93	BRAISED SPARERIBS
	L 93.01	SPARERIBS & SAUERKRAUT
	L 94	SWEET & SOUR SPARERIBS
	L 95	CANTONESE SPARERIBS
2	L 96	ROAST FRESH HAM
1	L 97	BARBECUED PORK LOIN
1	L 97.01	BARBECUED PORK LOIN (Prep Barbecue Sce)
3	L 98	SCRAPPLE
1	L 99	PORK ADOBO
	L100	SIMMERED PORK HOCKS (Ham Hocks)
	L100.01	SIMMERED PIGS' FEET
	L101	ITALIAN STYLE VEAL STEAKS
	L102	VEAL PAPRIKA STEAKS
	L103	VEAL STEAKS
	L103.01	VEAL PARMESAN
3	L105	VEAL CUBES PARMESAN
2c	L106	ROAST VEAL
2c	L106.01	ROAST VEAL W/HERBS
2	L107	BRAISED LIVER & ONIONS
2	L107.01	GRILLED LIVER
2	L108	BREADED LIVER
2	L108.01	BREADED LIVER W/ONION & MUSHROOM GRAVY
2	L109	LIVER FIESTA
1	L110	CORNED BEEF HASH
1	L110.01	HASH, CORNED BEEF (Cnd)
	L111	NEW ENGLAND BOILED DINNER
3	L112	SIMMERED CORNED BEEF
3	L112.01	APPLE GLAZED CORNED BEEF
	L113	BAKED FRANKFURTERS W/SAUERKRAUT
	L113.01	BAKED KNOCKWURST W/SAUERKRAUT
2	L115	FRANKFURTERS, CHEESE & BACON
2	L115.01	FRANKFURTERS, CHEESE & BACON W/BARBECUE SCE
1	L116	SIMMERED FRANKFURTERS
1	L116.01	SIMMERED KNOCKWURST
1	L116.02	SIMMERED POLISH SAUSAGE
3	L117	OVEN COOKED BOLOGNA
3	L117.01	GRILLED BOLOGNA
3	L117.02	GRILLED LUNCHEON MEAT
3	L117.03	GRILLED SALAMI
3	L117.04	OVEN COOKED LUNCHEON MEAT
3	L117.05	OVEN COOKED SALAMI
1	L118	FRIED RABBIT
1	L118.01	FRIED MARINATED RABBIT
2	L119	BAKED FISH
2	L119.01	BAKED FISH W/GARLIC BUTTER
2	L119.02	ONION-LEMON BAKED FISH
2	L119.03	SEASONED BAKED FISH
2	L119.04	LEMON BAKED FISH

2	L119.05	HERB BAKED FISH
1	L120	BAKED STUFFED FISH
2	L121	DEEP FAT FRIED FISH
2	L121.01	TEMPURA FRIED FISH
	L122	FRIED FISH
	L123	OVEN FRIED FISH
	L123.01	OVEN FRIED FISH (Dehy Fish Squares)
1	L124	BAKED FISH PORTIONS
1	L124.01	BAKED FISH PORTIONS (Batter Dipped)
1	L124.02	BAKED FISH STICKS
	L125	CHIPPER PERCH
1	L126	FRIED OYSTERS
1	L126.01	FRIED OYSTERS (Frz, Breaded)
1	L127	BOILED LOBSTER
1	L127.01	BOILED CRAWFISH (Spiney Lobster Tail)
1	L127.02	BOILED LOBSTER (Frz Whole Lobster)
1	L127.03	BOILED KING CRAB LEGS (Crab Legs, Alaskan King, Frz)
1	L128	SALMON CAKES
2	L129	SALMON LOAF
2	L129.01	TUNA LOAF
2	L129.02	CHICKEN LOAF
	L130	SCALLOPED SALMON & PEAS
	L130.01	SCALLOPED TUNA & PEAS
3	L131	CHOPSTICK TUNA
	L132	TUNA SALAD
	L132.01	SALMON SALAD
	L133	BAKED TUNA & NOODLES
	L133.01	BAKED TUNA & NOODLES (Soup, Cond, Cream of Mushroom)
	L134	FRIED SCALLOPS
	L134.01	FRIED SCALLOPS (Scallops, Frz, Breaded)
1	L135	CREOLE SCALLOPS
1	L135.01	CREOLE FISH
1	L135.02	CREOLE FISH FILLETS
3	L136	CREOLE SHRIMP
1	L137	FRENCH FRIED SHRIMP
1	L137.01	TEMPURA FRIED SHRIMP
1	L137.02	FRENCH FRIED SHRIMP (Frz, Breaded)
2	L138	SHRIMP CURRY
	L139	SHRIMP SALAD
	L140	SEAFOOD NEWBURG
	L141	CRAB CAKES
2	L142	HONEY GLAZED ROCK CORNISH HENS
2	L142.01	ROCK CORNISH HENS W/SYRUP GLAZE
2	L143	BAKED CHICKEN
3	L144	BAKED TURKEY & NOODLES
3	L144.01	BAKED CHICKEN & NOODLES (Chicken, Cnd)
3	L144.02	BAKED CHICKEN & NOODLES (Chicken, Dehy)
3	L144.03	BAKED TURKEY & NOODLES (Turkey, Cnd)
3	L144.04	BAKED TURKEY & NOODLES (Turkey, Boneless, Frz, Raw)
2	L145	CHICKEN VEGA
1	L146	BARBECUED CHICKEN
1	L146.01	BARBECUED CHICKEN (Prep Barbecue Sce)

Armed Forces Recipe Service

2	L147	CHICKEN A LA KING
2	L147.01	CHICKEN A LA KING (Chicken, Cnd)
2	L147.02	CHICKEN A LA KING (Soup, Cond, Cream of Chicken)
2	L147.03	CREAMED CHICKEN
2	L147.04	CREAMED TURKEY (Turkey, Boneless, Frz, Cooked)
2	L147.05	CREAMED TURKEY (Turkey, Boneless, Frz, Raw)
2	L147.06	CREAMED TURKEY (Turkey, Frz, Ready-to-Cook)
2	L147.07	TURKEY A LA KING (Turkey, Boneless, Frz, Cooked)
2	L147.08	TURKEY A LA KING (Turkey, Boneless, Frz, Raw)
2	L147.09	TURKEY A LA KING (Turkey, Frz, Ready-to-Cook)
2	L148	CHICKEN CACCIATORE
2	L148.01	CHICKEN CACCIATORE (Sce Mix, Tomato Basic)
2	L149	BAKED CHICKEN & GRAVY
2	L149.01	BAKED CHICKEN W/MUSHROOM GRAVY
2	L149.02	BAKED CHICKEN W/MUSHROOM GRAVY (Soup, Cond, Cream of Mushroom)
2	L149.03	BAKED GOLDEN CHICKEN (Soup, Cond, Cream of Chicken)
2	L150	CHICKEN POT PIE
2	L150.01	CHICKEN POT PIE (Chicken, Cnd)
2	L150.02	CHICKEN POT PIE (Chicken, Dehy)
2	L150.03	TURKEY POT PIE (Turkey, Boneless, Frz, Cooked)
2	L150.04	TURKEY POT PIE (Turkey, Boneless, Frz, Raw)
2	L150.05	TURKEY POT PIE (Turkey, Cnd)
2	L150.06	TURKEY POT PIE (Turkey, Frz, Ready-to-Cook)
2	L151	CHICKEN SALAD
2	L151.01	CHICKEN SALAD (Chicken, Cnd)
2	L151.02	TURKEY SALAD (Turkey, Boneless, Frz, Cooked)
2	L151.03	TURKEY SALAD (Turkey, Boneless, Frz, Raw)
2	L151.04	TURKEY SALAD (Turkey, Cnd)
	L152	CHICKEN TETRAZZINI (Chicken, Cnd)
	L152.01	CHICKEN TETRAZZINI (Chicken, Dehy)
	L152.02	CHICKEN TETRAZZINI (Soup, Cond, Cream of Chicken)
2	L153	COUNTRY STYLE CHICKEN (Maryland Fried)
1	L154	CREOLE CHICKEN
2	L155	FRIED CHICKEN
2	L155.01	SOUTHERN FRIED CHICKEN
2	L156	OVEN FRIED CHICKEN
2	L156.01	OVEN FRIED CHICKEN (Corn Flake Crumbs)
2	L157	PINEAPPLE CHICKEN
2c	L158	SAVORY BAKED CHICKEN
2	L159	NEWPORT FRIED CHICKEN
2	L160	CHICKEN CHOW MEIN
2	L160.01	CHICKEN CHOW MEIN (Chicken, Cnd)
2	L160.02	TURKEY CHOW MEIN (Turkey, Cnd)
2	L160.03	TURKEY CHOW MEIN (Turkey, Boneless, Frz, Cooked)
2	L160.04	TURKEY CHOW MEIN (Turkey, Boneless, Frz, Raw)
2	L160.05	TURKEY CHOW MEIN (Turkey, Frz, Ready-to-Cook)
2	L161	ROAST TURKEY (Turkey, Frz, Ready-to-Cook)
2	L162	ROAST TURKEY (Turkey, Boneless)
2	L162.01	ROAST TURKEY (Turkey, Boneless, Frz, Cooked)
	L163	TURKEY NUGGETS (Scallops)
1	L164	ROAST DUCK
1	L164.01	HAWAIIAN BAKED DUCK

1	L164.02	ROAST DUCK W/APPLE JELLY GLAZE
1	L164.03	HONEY GLAZED DUCK
3	L165	PIZZA
3	L165.01	PIZZA (Thick Crust)
3	L165.02	MUSHROOM, GREEN PEPPER & ONION PIZZA
3	L165.03	HAMBURGER PIZZA
3	L165.04	PEPPERONI, GREEN PEPPER & MUSHROOM PIZZA
3	L165.05	PEPPERONI PIZZA
3	L165.06	PIZZA (Roll Mix)
3	L165.07	PORK OR ITALIAN SAUSAGE PIZZA
3	L165.08	FRENCH BREAD PIZZA
3	L165.09	PORK OR ITALIAN SAUSAGE, GREEN PEPPER & ONION PIZZA
3	L166	PIZZA (12 Inch Frz Crusts)
3	L166.01	PIZZA (Cheese, Prep, Frz)
3	L166.02	FAST FOOD PIZZA
1	L167	FRENCH FRIED FISH PORTIONS
1	L167.01	FRENCH FRIED FISH PORTIONS (Batter Dipped)
1	L167.02	FRENCH FRIED FISH STICKS
1	L167.03	FISH & CHIPS (Batter Dipped)
1	L171	CHUCK WAGON STEW (Beans w/Beef)
1	L172	BEEF STEW (Cnd Beef Chunks w/Natural Jces)
1	L173	STEAK RANCHERO
1	L174	FRIED CHICKEN (Fast Food)
1	L174.01	FRIED CHICKEN (Preckd, Breaded Chicken, Frz for Deep Fat Fryer)
1	L174.02	FRIED CHICKEN (Preckd, Breaded Chicken, Frz for Oven)
1	L175	BAKED CHICKEN & RICE
1	L175.01	BAKED CHICKEN & RICE (Chicken, Cnd)
1	L175.02	BAKED CHICKEN & RICE (Chicken, Dehy)
1	L176	SOUTHERN FRIED CATFISH FILLETS
1	L176.01	SOUTHERN FRIED WHOLE CATFISH (Whole Dressed Catfish)
1	L177	TURKEY CUTLET (Turkey, Boneless, Frz, Cooked)
1	L177.01	TURKEY CUTLET (Turkey, Boneless, Frz, Raw)
1	L178	GRILLED HAMBURGER STEAK
1	L178.01	GRILLED MOCK FILLET STEAK
1	L179	BAKED WHOLE TROUT
1	L179.01	BAKED TROUT FILLETS
1	L181	DEVILED CRAB
1	L182	GROUND BEEF CORDON BLEU
1	L183	SHRIMP SCAMPI
1	L184	JAEGERSCHNITZEL (Veal Steak w/Mushroom Sce)
1	L185	BEEF RAVIOLI
1	L185.01	CHEESE RAVIOLI
1	L185.02	MEAT RAVIOLI (Ravioli, Meat, in Tomato Sce, Cnd)
2	L186	MACARONI-TUNA SALAD
2	L187	CHALUPA
2	L188	SZECHWAN CHICKEN
2	L189	JAMBALAYA
2	L190	CHINESE FIVE-SPICE CHICKEN
2	L191	SPICY BAKED FISH
2	L192	TERIYAKI CHICKEN
3	L193	OVEN FRIED CHICKEN FILLETS (3 ounces)

3	L193.01		DEEP FAT FRIED CHICKEN FILLETS (3 ounces)
3	L193.02		OVEN FRIED CHICKEN FILLETS (5 ounces)
3	L193.03		DEEP FAT FRIED CHICKEN FILLETS (5 ounces)
3	L193.04		OVEN FRIED CHICKEN FILLET NUGGETS
3	L193.05		DEEP FAT FRIED CHICKEN FILLET NUGGETS
3	M	1	APPLE, CELERY, & PINEAPPLE SALAD
2	M	2	SPINACH SALAD
2	M	3	CABBAGE, APPLE & CELERY SALAD
2	M	3.01	CABBAGE, APPLE, & RAISIN SALAD
3	M	4	FRIJOLE SALAD
3	M	5	CARROT SALAD
3	M	5.01	CARROT & PINEAPPLE SALAD
3	M	5.02	CARROT & RAISIN SALAD
3	M	5.03	CARROT, CELERY, & APPLE SALAD
3	M	5.04	CARROT, RAISIN, & CELERY SALAD
3	M	6	STUFFED CELERY
3	M	6.01	BLUE CHEESE STUFFED CELERY
3	M	6.02	COTTAGE CHEESE STUFFED CELERY
3	M	6.03	COTTAGE CHEESE & RELISH STUFFED CELERY
3	M	6.04	PEANUT BUTTER STUFFED CELERY
3	M	6.05	CREAM CHEESE STUFFED CELERY
2	M	7	CHEF'S SALAD
2	M	7.01	CHEF'S SALAD ITALIAN STYLE
2	M	7.02	CHEF'S SALAD W/CROUTONS
3	M	8	COLE SLAW
3	M	8.01	MEXICAN COLE SLAW
2	M	9	COLE SLAW W/CREAMY DRESSING
2	M	9.01	COLE SLAW W/VINEGAR DRESSING
2	M	9.02	CABBAGE & CARROT SLAW W/CREAMY DRESSING
2	M	9.03	PINEAPPLE COLE SLAW
2	M	9.04	PINEAPPLE MARSHMALLOW COLE SLAW
2	M	9.05	VEGETABLE SLAW W/CREAMY DRESSING
3	M	10	COLE SLAW (Dehy Comp Cabbage)
3	M	12	COTTAGE CHEESE SALAD
2	M	13	COTTAGE CHEESE & PEACH SALAD
2	M	13.01	COTTAGE CHEESE & APRICOT SALAD
2	M	13.02	COTTAGE CHEESE & PEAR SALAD
2	M	13.03	COTTAGE CHEESE & PINEAPPLE SALAD
2	M	13.04	COTTAGE CHEESE & PEACH SLICE SALAD
2	M	14	COTTAGE CHEESE & TOMATO SALAD
2	M	14.01	COTTAGE CHEESE & SLICED TOMATO SALAD
3	M	15	CUCUMBER & ONION SALAD
2	M	16	CUCUMBER & SOUR CREAM SALAD
2	M	16.01	CUCUMBER, ONION & SOUR CREAM SALAD
2	M	16.02	CUCUMBER, ONION, TOMATO & SOUR CREAM SALAD
2	M	17	FRUIT SALAD
3	M	18	GARDEN COTTAGE CHEESE SALAD
3	M	19	GARDEN VEGETABLE SALAD
3	M	20	MARINATED CARROTS
2	M	22	JELLIED BANANA SALAD
	M	23	JELLIED CRANBERRY & ORANGE SALAD
	M	23.01	JELLIED CRANBERRY & ORANGE SALAD (Cnd Cranberry Sce)

3	M 24	JELLIED CRANBERRY & PINEAPPLE SALAD
3	M 25	JELLIED FRUIT SALAD
3	M 25.01	JELLIED ORANGE SALAD
3	M 25.02	JELLIED PEAR SALAD
3	M 25.03	JELLIED PINEAPPLE, PEAR, & BANANA SALAD
3	M 25.04	JELLIED STRAWBERRY SALAD
3	M 26	JELLIED FRUIT COCKTAIL SALAD
3	M 26.01	SHIMMERY FRUIT & VEGETABLE SALAD
3	M 27	GERMAN COLE SLAW
	M 28	JELLIED SPICED CHERRY SALAD
	M 28.01	JELLIED SPICED BLACK CHERRY SALAD
	M 28.02	JELLIED SPICED PEACH SALAD
3	M 29	ITALIAN STYLE PASTA SALAD
3	M 31	KIDNEY BEAN SALAD
2	M 32	FRUIT MEDLEY SALAD
2	M 33	LETTUCE & TOMATO SALAD
2	M 33.01	LETTUCE WEDGE SALAD
1	M 34	MACARONI SALAD
2	M 35	MIXED FRUIT SALAD
3	M 36	PERFECTION SALAD
3	M 36.01	GOLDEN GLOW SALAD
3	M 36.02	JELLIED SPRING SALAD
	M 37	PICKLED BEET & ONION SALAD
2	M 40	POTATO SALAD
2	M 40.01	DEVILED POTATO SALAD
2	M 40.02	POTATO SALAD W/VINEGAR DRESSING
2	M 41	POTATO SALAD (Potatoes, Dehy, Sliced)
2	M 41.01	DEVILED POTATO SALAD (Potatoes, Dehy, Sliced)
2	M 42	HOT POTATO SALAD
2	M 43	HOT POTATO SALAD (Potatoes, Dehy, Sliced)
2	M 43.01	HOT POTATO SALAD (Potatoes, Dehy, Diced)
3	M 44	SPRING SALAD
1	M 45	THREE BEAN SALAD
1	M 45.01	PICKLED GREEN BEAN SALAD
1	M 45.02	PICKLED GREEN BEAN SALAD (Dehy, Comp Green Beans)
3	M 46	TOSSED LETTUCE, CUCUMBER, & TOMATO SALAD
3	M 47	TOSSED GREEN SALAD
2	M 48	TOSSED VEGETABLE SALAD
3	M 49	VEGETABLE SALAD
2	M 50	WALDORF SALAD
2	M 50.01	APPLE, CELERY, & DATE SALAD
2	M 50.02	APPLE, CELERY, & RAISIN SALAD
3	M 52	GUACAMOLE
2	M 53	GERMAN STYLE TOMATO SALAD
2	M 53.01	COUNTRY STYLE TOMATO SALAD
3	M 54	BACON-SOUR CREAM DRESSING
3	M 55	CELERY SEED DRESSING
3	M 56	CREAMY FRUIT DRESSING
3	M 56.01	QUICK FRUIT DRESSING
2	M 57	ZERO SALAD DRESSING
3	M 58	FRENCH DRESSING
3	M 58.01	LOW CALORIE FRENCH DRESSING

3	M	59	BLUE CHEESE DRESSING
3	M	60	GARLIC FRENCH DRESSING
3	M	61	TANGY SALAD DRESSING
3	M	62	CHIFFONADE DRESSING
3	M	63	MAYONNAISE
3	M	64	CREAMY ITALIAN DRESSING
3	M	65	CREAMY HORSERADISH DRESSING
3	M	66	LOW CALORIE TOMATO DRESSING
3	M	67	RUSSIAN DRESSING
3	M	68	SOUR CREAM DRESSING
3	M	68.01	BLUE CHEESE & SOUR CREAM DRESSING
3	M	69	VINEGAR & OIL DRESSING
3	M	70	THOUSAND ISLAND DRESSING
3	M	71	VINAIGRETTE DRESSING
3	M	72	TOMATO FRENCH DRESSING
	M	75	CRANBERRY ORANGE RELISH
	M	75.01	CRANBERRY, ORANGE, & APPLE RELISH
3	M	76	CORN RELISH
2	M	77	PASTA SALAD
2	M	78	COBB SALAD
2	M	79	TACO SALAD
2	M	85	LOW CALORIE THOUSAND ISLAND DRESSING
3	N	1	TOASTED BACON, LETTUCE, & TOMATO SANDWICH
3	N	1.01	BACON, CHEESE, LETTUCE, & TOMATO SANDWICH
3	N	1.02	BACON, LETTUCE & TOMATO SANDWICH
1	N	1.03	TOASTED BACON, CHEESE, LETTUCE & TOMATO SANDWICH
1	N	3	STEAK & CHEESE SUBMARINE
1	N	3.01	STEAK, CHEESE & ONION SUBMARINE
1	N	3.02	STEAK & ONION SUBMARINE
3	N	4	ROAST BEEF SANDWICH
3	N	4.01	ROAST PORK SANDWICH
3	N	4.02	ROAST TURKEY SANDWICH
3	N	4.03	ROAST VEAL SANDWICH
3	N	4.04	ROAST TURKEY CROISSANT W/CHUTNEY SCE
3	N	4.05	ROAST BEEF CROISSANT
3	N	4.06	TURKEY CROISSANT
2	N	6	GRILLED CHEESE SANDWICH
2	N	6.01	GERMAN STYLE HAMWICH
2	N	6.02	GRILLED CHEESE & BACON SANDWICH
2	N	6.03	GRILLED CHEESE & HAM SANDWICH
2	N	6.04	OVEN TOASTED CHEESE SANDWICH
2	N	6.05	OVEN TOASTED GERMAN STYLE HAMWICH
2	N	6.06	OVEN TOASTED CHEESE & BACON SANDWICH
2	N	6.07	OVEN TOASTED CHEESE & HAM SANDWICH
1	N	8	CHICKEN SALAD SANDWICH
1	N	8.01	TURKEY SALAD SANDWICH
1	N	8.02	CHICKEN SALAD SUBMARINE
3	N	9	SLICED CORNED BEEF SANDWICH
3	N	9.01	CORNED BEEF & CHEESE SANDWICH
3	N	9.02	HOT CORNED BEEF SANDWICH
1	N	10	EGG SALAD SANDWICH
1	N	10.01	EGG & HAM SALAD SANDWICH

1	N	10.02	NEW YORK EGG SALAD SANDWICH (Egg & Tomato)
2	N	11	HAM SANDWICH
2	N	11.01	FRIED HAM SANDWICH
2	N	11.02	HAM & CHEESE SANDWICH
2	N	11.03	HAM & TOMATO SANDWICH
	N	12	DEVILED HAM SANDWICH
	N	12.01	DEVILED HAM & TOMATO SANDWICH
	N	12.02	DEVILED HAM & TOMATO HOAGIE
1	N	13	HAM SALAD SANDWICH
1	N	13.01	HAM & CHEESE SALAD SANDWICH
1	N	13.02	HAM SALAD SUBMARINE
3	N	14	PEANUT BUTTER & JELLY SANDWICH
3	N	14.01	PEANUT BUTTER & JAM SANDWICH
1	N	15	TUNA SALAD SANDWICH
1	N	15.01	GRILLED TUNA & CHEESE SANDWICH
1	N	15.02	SALMON SALAD SANDWICH
1	N	15.03	TUNA & TOMATO SANDWICH
1	N	15.04	TUNA SALAD SUBMARINE
3	N	16	CREAM CHEESE BAGEL
3	N	16.01	CREAM CHEESE & TOMATO BAGEL
3	N	16.02	CREAM CHEESE & OLIVE BAGEL
2	N	17	COLD CUT SANDWICH
2	N	17.01	COLD CUT SANDWICH W/CHEESE
	N	18	WESTERN SANDWICH (Denver)
2	N	19	SUBMARINE SANDWICH
2	N	19.01	ITALIAN STYLE SUBMARINE
2	N	20	GRILLED REUBEN SANDWICH
2	N	20.01	OVEN-TOASTED REUBEN SANDWICH
2	N	20.02	GRILLED REUBEN FRANKWICH
2	N	20.03	GRILLED REUBEN PASTRAMI SANDWICH
1	N	21	TACOS
1	N	21.01	TACOS (Seasoning Mix, Taco)
1	N	21.02	BURRITOS (Beef & Bean, Frz)
1	N	22	CANNONBALL SANDWICH (Meatball)
1	N	22.01	HOT ITALIAN SAUSAGE SANDWICH
1	N	22.02	CANNONBALL SANDWICH (Cnd Meatballs in Tomato Sce)
1	N	22.03	CHEESY CANNONBALL SANDWICH
	N	23	HOT PASTRAMI SANDWICH
2	N	24	BARBECUED BEEF SANDWICH (Cnd Beef Chunks w/Natural Jces)
2	N	25	MONTE CRISTO SANDWICH
2	N	26	CHILI SIZE
2	N	26.01	CHILI SIZE W/BEANS
2	N	26.02	CHILI SIZE W/BEANS (Chili Con Carne)
1	N	27	BARBECUED BEEF SANDWICH (Sloppy Joe)
1	N	27.01	BARBECUED PORK SANDWICH (Pork Butt)
1	N	27.02	BARBECUED PORK SANDWICH (Cnd Pork w/Natural Jces)
1	N	27.03	BARBECUED PORK SANDWICH (Frz Barbecued Pork)
1	N	27.04	BARBECUED BEEF SANDWICH (Beef w/Barbecue Sce)
1	N	28	ITALIAN PEPPER BEEF SANDWICH
1	N	28.01	BEEF & ONION SANDWICH
1	N	28.02	ITALIAN BEEF PEPPER SANDWICH (Beef, Top Round Roast, Frz)
1	N	28.03	PEPPER STEAK SANDWICH

3	N	29	GRILLED HAMBURGER (Beef Patties w/Soy Protein)
3	N	29.01	DELUXE HAMBURGER (Beef Patties w/Soy Protein)
3	N	29.02	CHEESEBURGER (Beef Patties w/Soy Protein)
3	N	29.03	CHEESY BACON BURGER (Beef Patties w/Soy Protein)
3	N	29.04	DOUBLE DECKER CHEESEBURGER (Beef Patties w/Soy Protein)
3	N	29.05	CHILIBURGER (Beef Patties w/Soy Protein)
3	N	29.06	JALAPENO CHILIBURGER (Beef Patties w/Soy Protein)
3	N	29.07	PIZZABURGER (Beef Patties w/Soy Protein)
3	N	29.08	JALAPENO CHILI CHEESEBURGER (Beef Patties w/Soy Protein)
3	N	29.09	DELUXE CHEESEBURGER (Beef Patties w/Soy Protein)
3	N	30	SIMMERED FRANKFURTER ON ROLL
3	N	30.01	CHILI DOG
3	N	30.02	CHILI DOG W/CHEESE & ONIONS
3	N	30.03	SIMMERED QUARTER POUND FRANKFURTER
3	N	30.04	GRILLED FRANKFURTER ON ROLL
3	N	30.05	GRILLED FRANKFURTER W/FRIED PEPPERS & ONIONS
1	N	31	SUPER-BURGER
1	N	32	FISHWICH
1	N	32.01	CHEESE FISHWICH
1	N	32.02	CHEESE FISHWICH (Batter Dipped Fish Portions)
1	N	32.03	FISHWICH (Batter Dipped Fish Portions)
	N	33	HOT ROAST TURKEY SANDWICH
3	N	35	HOT ROAST BEEF SANDWICH (Oven Roast)
3	N	35.01	HOT ROAST BEEF SANDWICH (Pot Roast)
3	N	35.02	HOT ROAST BEEF SANDWICH (Preckd Roast Beef)
	N	36	HOT ROAST PORK SANDWICH
	N	36.01	HOT ROAST PORK SANDWICH (Fresh Ham)
2	N	37	GRILLED HAM, EGG, & CHEESE SANDWICH
2	N	37.01	GRILLED BACON, EGG, & CHEESE SANDWICH
2	N	37.02	GRILLED HAM & EGG SANDWICH
2	N	37.03	GRILLED SAUSAGE, EGG, & CHEESE SANDWICH
2	N	38	JIMBO
2	N	38.01	GRILLED JIMBO
2	N	39	HAMBURGER HERO SANDWICH
2	N	40	TACO BURGER
3	N	42	CORN DOG
3	N	42.01	CORN DOG (Corn Bread Mix)
3	N	42.02	CORN DOG (Frz)
	N	43	MONTE CARLO SANDWICH (Open-Faced Turkey & Ham)
2	N	44	ITALIAN VEAL CUTLET SUBMARINE SANDWICH
2	N	44.01	ITALIAN VEAL CUTLET SUBMARINE (Cnd Pizza Sce)
3	N	45	OVEN FRIED CHICKEN FILLET SANDWICHES
3	N	45.01	DEEP FAT FRIED CHICKEN FILLET SANDWICHES
3	N	45.02	CHICKEN FILLET & CHEESE SANDWICHES
3	N	46	GRILLED HAMBURGER (Beef Patties)
3	N	46.01	CHEESEBURGER (Beef Patties)
3	N	46.02	CHEESY BACON BURGER (Beef Patties)
3	N	46.03	DOUBLE DECKER CHEESEBURGER (Beef Patties)
3	N	46.04	CHILIBURGER (Beef Patties)
3	N	46.05	JALAPENO CHILIBURGER (Beef Patties)
3	N	46.06	PIZZABURGER (Beef Patties)
3	N	46.07	JALAPENO CHILI CHEESEBURGER (Beef Patties)

3	N	46.08	DELUXE HAMBURGER (Beef Patties)
3	N	46.09	DELUXE CHEESEBURGER (Beef Patties)
1	N	47	BEEF & BEAN TOSTADAS
1	N	47.01	BEEF & BEAN TOSTADAS W/GUACAMOLE
1	N	48	ENGLISH MUFFIN W/BACON, EGG, & CHEESE
1	N	48.01	ENGLISH MUFFIN W/HAM, EGG, & CHEESE
1	N	48.02	ENGLISH MUFFIN W/BACON, HAM, & CHEESE
1	N	48.03	ENGLISH MUFFIN W/SAUSAGE, EGG, & CHEESE
1	N	48.04	ENGLISH MUFFIN W/PEANUT BUTTER & JELLY OR JAM
1	N	48.05	ENGLISH MUFFIN W/PEANUT BUTTER & HONEY
1	N	48.06	ENGLISH MUFFIN W/CANADIAN BACON, EGG, & CHEESE
3	N	49	GRILLED POLISH SAUSAGE SANDWICH W/SAUERKRAUT, ONIONS & MUSTARD
3	N	49.01	GRILLED POLISH SAUSAGE SANDWICH
3	N	49.02	POLISH SAUSAGE, SAUERKRAUT & SWISS CHEESE
3	N	49.03	GRILLED BOCKWURST ON ROLLS
3	N	49.04	GRILLED BRATWURST ON ROLLS
3	N	49.05	SIMMERED KNOCKWURST ON ROLLS
2	N	50	GYROS
3	O	1	WHITE SCE
3	O	1.01	CHEESE SCE
3	O	2	BARBECUE SCE
3	O	2.01	BARBECUE SCE (Prep)
2	O	3	CHERRY SCE (For Meat)
2	O	3.01	CHERRY SCE (Pregelatinized Starch)
2	O	3.02	CHERRY SCE (Pie Filling, Prep)
3	O	5	CREOLE SCE
3	O	5.01	SPANISH SCE
3	O	6	HOT MUSTARD SCE
1	O	7	LEMON BUTTER SCE
2	O	8	SWEET SOUR SCE
3	O	9	PINEAPPLE SCE (For Ham)
3	O	9.01	RAISIN SCE (For Ham)
1	O	11	SEAFOOD COCKTAIL SCE
1	O	11.01	SEAFOOD COCKTAIL SCE (Seafood Cocktail Sce, Prep)
3	O	12	PIZZA SCE
3	O	12.01	PIZZA SCE (Cnd)
3	O	13	TARTAR SCE
2	O	14	TERIYAKI SCE
2	O	14.01	TERIYAKI SCE (Prep)
2	O	15	TOMATO SCE
2	O	16	BROWN GRAVY
2	O	16.01	BROWN GRAVY (Brown Gravy Mix)
2	O	16.02	CHICKEN GRAVY
2	O	16.03	CHILI GRAVY
2	O	16.04	GIBLET GRAVY
2	O	16.05	MUSHROOM GRAVY
2	O	16.06	ONION GRAVY
2	O	16.07	QUICK ONION GRAVY (Soup, Dehy, Onion)
2	O	16.08	TURKEY GRAVY
2	O	16.09	VEGETABLE GRAVY
2	O	16.10	ONION & MUSHROOM GRAVY
1	O	17	CREAM GRAVY

Armed Forces Recipe Service

1	O	17.01	CREAM ONION GRAVY
3	O	18	NATURAL PAN GRAVY (Au Jus)
1	O	19	TOMATO GRAVY
3	O	20	CORN BREAD DRESSING
3	O	21	SAVORY BREAD DRESSING
3	O	21.01	APPLE DRESSING
3	O	21.02	GIBLET DRESSING
3	O	21.03	OYSTER DRESSING
3	O	21.04	SAUSAGE DRESSING
3	O	22	CHINESE MUSTARD SCE
1	O	23	HORSERADISH SCE
3	O	25	TACO SCE
3	O	25.01	TACO SCE (Prep)
3	O	25.02	SALSA
2	O	26	SZECHWAN SCE
1	PG	1	GUIDELINES FOR CND, COND SOUPS
	P	1	BEEF STOCK
	P	1.01	BEEF BARLEY SOUP
	P	1.02	BEEF NOODLE SOUP
	P	1.03	BEEF RICE SOUP
	P	2	TURKEY STOCK
	P	2.01	TURKEY OR CHICKEN NOODLE SOUP
	P	2.02	TURKEY OR CHICKEN RICE SOUP
3	P	3	CREOLE SOUP
2	P	4	ONION SOUP
2	P	4.01	FRENCH ONION SOUP
2	P	5	TOMATO BOUILLON
3	P	6	TOMATO SOUP
3	P	6.01	TOMATO RICE SOUP
1	P	7	VEGETABLE SOUP
1	P	7.01	MINESTRONE SOUP
3	P	8	NAVY BEAN SOUP
3	P	8.01	BEAN SOUP W/SMOKED,CURED HAM HOCKS
3	P	8.02	KNICKERBOCKER SOUP (Bean, Tomato, & Bacon)
3	P	8.03	OLD FASHIONED BEAN SOUP
	P	9	CHICKEN EGG DROP SOUP
	P	10	CHICKEN GUMBO SOUP
	P	10.01	SHRIMP GUMBO
3	P	11	CORN CHOWDER
3	P	11.01	CHICKEN CORN CHOWDER
3	P	12	MANHATTAN FISH CHOWDER
3	P	12.01	MANHATTAN CLAM CHOWDER
3	P	13	NEW ENGLAND FISH CHOWDER
3	P	13.01	NEW ENGLAND CLAM CHOWDER
1	P	14	CREAM OF MUSHROOM SOUP
3	P	15	CREAM OF POTATO SOUP (Potatoes, Dehy, Sliced)
3	P	15.01	CREAM OF POTATO SOUP (Potatoes, White, Fresh)
3	P	16	CREAM OF POTATO SOUP (Potatoes, Instnt)
1	P	17	SPANISH SOUP (Soup, Dehy, Onion)
1	P	17.01	ONION SOUP (Soup, Dehy, Onion)
1	P	17.02	ONION CLAM GUMBO (Soup, Dehy, Onion)
1	P	17.03	MEXICAN ONION CORN SOUP (Soup, Dehy, Onion)

3	P	18	TOMATO-VEGETABLE SOUP (Dehy)
3	P	18.01	BEEF-NOODLE SOUP (Dehy)
3	P	18.02	CHICKEN-NOODLE SOUP (Dehy)
3	P	18.03	CHICKEN-NOODLE VEGETABLE SOUP (Dehy)
1	P	19	PEPPER POT SOUP
	P	20	CHICKEN VEGETABLE (Mulligatawny) SOUP
1	P	21	OYSTER STEW
1	P	21.01	SCALLOP STEW SUPREME
	P	22.01	CHICKEN-MUSHROOM SOUP (Cnd Cond Soup Mergers)
	P	22.02	DOUBLY GOOD CHICKEN SOUP (Cnd Cond Soup Mergers)
	P	22.03	LOGGING SOUP (Cnd Cond Soup Mergers)
	P	22.04	TOMATO-NOODLE SOUP (Cnd Cond Soup Mergers)
	P	22.05	VEGETABLE SUPREME SOUP (Cnd Cond Soup Mergers)
2	P	23	SPLIT PEA SOUP
2	P	23.01	PUREE MONGOLE (Soup, Cond, Tomato)
2	P	23.02	SPLIT PEA SOUP W/SMOKED CURED PORK HOCKS (Ham Hocks)
1	QG	1	GUIDELINES FOR HEATING CND VEGETABLES
3	QG	2	GUIDELINES FOR SIMMERING FRESH VEGETABLES
1	QG	3	GUIDELINES FOR COOKING FRZ VEGETABLES
1	QG	5	GUIDELINES FOR HEATING DEHY, COMP VEGETABLES
1	QG	6	GUIDELINES FOR STEAM COOKING VEGETABLES
	Q	1	CREAMED ASPARAGUS (Frz)
	Q	1.01	ASPARAGUS AU GRATIN
	Q	1.02	ASPARAGUS, CREAMED (Cnd)
3	Q	2	BAKED BEANS (Cnd Beans)
3	Q	2.01	BAKED BEANS (Kidney Beans, Cnd)
3	Q	2.02	BAKED BEANS (Pinto Beans, Cnd)
2	Q	3	BOSTON BAKED BEANS
2	Q	3.01	SAVORY BAKED BEANS
3	Q	4	ITALIAN-STYLE BAKED BEANS
3	Q	4.01	ITALIAN-STYLE BAKED BEANS (Kidney Beans, Cnd)
3	Q	4.02	ITALIAN-STYLE BAKED BEANS (Pinto Beans, Cnd)
1	Q	5	SIMMERED DRY BEANS (Kidney, Lima, Pinto, & White)
1	Q	5.01	SAVORY STYLE BEANS (Kidney, Lima, Pinto, & White)
1	Q	5.02	SIMMERED DRY BEANS (Wo/Bacon)
1	Q	6	SPANISH STYLE LIMA BEANS
2	Q	7	LYONNAISE GREEN OR WAX BEANS
2	Q	7.01	GREEN BEANS CREOLE
2	Q	7.02	GREEN BEANS W/MUSHROOMS
2	Q	7.03	GREEN BEANS NICOISE
2	Q	7.04	GREEN BEANS SOUTHERN STYLE
2	Q	8	HARVARD BEETS
2	Q	8.01	BEETS IN ORANGE-LEMON SCE
	Q	9	HOT SPICED BEETS
2	Q	10	BROCCOLI POLONAISE
2	Q	10.01	BRUSSELS SPROUTS POLONAISE
2	Q	10.02	CAULIFLOWER POLONAISE
	Q	11	SPROUTS SUPERBA
3	Q	12	FRIED CABBAGE
3	Q	12.01	CALICO CABBAGE
3	Q	12.02	FRIED CABBAGE W/BACON
3	Q	13	SEASONED CABBAGE (Dehy Comp Cabbage)

3	Q	13.01	BAVARIAN CABBAGE (Dehy Comp Cabbage)
3	Q	15	CHINESE FRIED CABBAGE (Dehy Comp Cabbage)
3	Q	15.01	CHINESE FRIED CABBAGE (Fresh Cabbage)
1	Q	16	CARROT & CELERY AMANDINE
	Q	17	LYONNAISE CARROTS
	Q	17.01	GLAZED CARROTS
	Q	17.02	GINGER GLAZED CARROTS
	Q	17.03	CARROTS NORMANDIE
2	Q	18	CAULIFLOWER AU GRATIN
2	Q	18.01	CURRIED CAU.OLIFLOWER W/PEAS
3	Q	19	GERMAN POTATO GRIDDLE CAKES (Potatoes, Dehy, Sliced)
3	Q	20	FRENCH FRIED CAULIFLOWER
3	Q	20.01	FRENCH FRIED OKRA
2	Q	21	CORN FRITTERS
2	Q	21.01	CORN FRITTERS (Pancake Mix)
	Q	22	CORN PUDDING
	Q	22.01	CORN PUDDING (Whole Kernel Corn)
	Q	22.02	SOUTHERN STYLE CORN
2	Q	23	SCALLOPED CREAM STYLE CORN
2	Q	23.01	SCALLOPED WHOLE KERNEL CORN (Cnd)
2	Q	23.02	SCALLOPED WHOLE KERNEL CORN (Frz)
	Q	24	BROCCOLI PARMESAN
	Q	24.01	BRUSSELS SPROUTS PARMESAN
	Q	24.02	CAULIFLOWER PARMESAN
1	Q	25	VEGETABLE STIR FRY
	Q	26	BAKED CORN & TOMATOES
1	Q	27	CALICO CORN
1	Q	27.01	CORN O'BRIEN
1	Q	27.02	MEXICAN CORN
3	Q	28	FRENCH FRIED EGGPLANT
3	Q	28.01	EGGPLANT PARMESAN
3	Q	28.02	EGGPLANT PARMESAN (Pizza Sce, Cnd)
3	Q	29	SOUTHERN STYLE GREENS (Fresh Collard)
3	Q	29.01	SOUTHERN STYLE GREENS (Frz Collard, Mustard, or Turnip Greens)
3	Q	29.02	SWEET SOUR GREENS
3	Q	29.03	SOUTHERN STYLE GREENS (Fresh Kale)
1	Q	30	SAUTEED MUSHROOMS
1	Q	30.01	SAUTEED MUSHROOMS & ONIONS
2	Q	31	OKRA & TOMATO GUMBO
	Q	32	SOUTHERN FRIED OKRA
2	Q	34	BAKED ONIONS W/TOMATOES
2	Q	34.01	SPANISH ONIONS
2	Q	35	FRENCH FRIED ONION RINGS (Flour Method)
2	Q	35.01	FRENCH FRIED ONION RINGS (Frz)
2	Q	35.02	TEMPURA FRIED ONION RINGS
2	Q	35.03	FRENCH FRIED ONION RINGS (Onion Ring Method)
3	Q	36	FRIED ONIONS
2	Q	37	SMOTHERED ONIONS (Dehy Onions)
	Q	38	FRIED PARSNIPS
3	Q	39	GREEN BEANS W/CORN (Frz)
3	Q	39.01	GREEN BEANS W/CORN (Cnd Green Beans)
3	Q	39.02	GREEN BEANS W/CORN (Dehy, Comp Green Beans)

2	Q 40	CREAMED PEAS
2	Q 41	PEAS (Frz) W/MUSHROOMS
2	Q 41.01	PEAS W/CARROTS
2	Q 41.02	PEAS W/CELERY
2	Q 41.03	PEAS W/ONIONS
2	Q 41.04	PEAS W/MUSHROOMS (Dehy Comp Peas)
2	Q 41.05	PEAS W/MUSHROOMS (Cnd Peas)
3	Q 42	GREEN BEANS PARISIENNE (Cnd)
3	Q 42.01	GREEN BEANS PARISIENNE (Frz)
3	Q 43	RED CABBAGE W/SWEET & SOUR SCE
1	Q 44	BAKED POTATOES
1	Q 44.01	QUICK BAKED POTATO HALVES
1	Q 45	FRENCH FRIED POTATOES
1	Q 45.01	FRENCH FRIED POTATOES (Frz)
1	Q 45.02	FRENCH FRIED POTATOES (Frz) Oven Method
1	Q 45.03	FRENCH FRIED SHOESTRING POTATOES (Frz)
1	Q 45.04	FRENCH FRIED SHOESTRING POTATOES (Frz) Oven Method
1	Q 45.05	FRENCH FRIED POTATOES (Dehy Potato Mix)
	Q 46	HASHED BROWN POTATOES
	Q 46.01	COTTAGE FRIED POTATOES
	Q 46.02	HASHED BROWN POTATOES (Frz, Shredded)
	Q 46.03	LYONNAISE POTATOES
	Q 47	HOME FRIED POTATOES
3	Q 48	MASHED POTATOES
3	Q 48.01	GRILLED POTATO PATTIES
1	Q 49	O'BRIEN POTATOES
1	Q 49.01	O'BRIEN POTATOES (Frz, Diced Potatoes)
	Q 50	OVEN BROWNED POTATOES
	Q 50.01	FRANCONIA POTATOES
	Q 50.02	FRENCH BAKED POTATOES
	Q 50.03	OVEN-GLO POTATOES
1	Q 51	POTATOES AU GRATIN (Potatoes, White, Fresh)
1	Q 51.01	POTATOES AU GRATIN (Potatoes, Dehy, Sliced)
1	Q 51.02	POTATOES AU GRATIN (Soup, Cond, Cream of Mushroom)
	Q 52	RISSOLE POTATOES
	Q 52.01	RISSOLE POTATOES (Alternate Method)
1	Q 53	SCALLOPED POTATOES
1	Q 53.01	SCALLOPED POTATOES & ONIONS
1	Q 53.02	SCALLOPED POTATOES SUPREME
1	Q 54	HASHED BROWN POTATOES (Potatoes, Dehy, Sliced)
1	Q 54.01	LYONNAISE POTATOES (Potatoes, Dehy, Sliced)
1	Q 54.02	O'BRIEN POTATOES (Potatoes, Dehy, Sliced)
1	Q 54.03	HASHED BROWN POTATOES (Potatoes, Dehy, Shredded [Hash Brown])
1	Q 54.04	HASHED BROWN POTATOES (Potatoes, Dehy, Diced)
1	Q 55	SCALLOPED POTATOES & ONIONS (Potatoes, Dehy, Sliced)
1	Q 55.01	SCALLOPED POTATOES (Potatoes, Dehy, Sliced)
1	Q 55.02	SCALLOPED POTATOES (Potatoes, Dehy, Diced)
	Q 56	GOLDEN POTATO BALLS (Potatoes, Instnt)
3	Q 57	MASHED POTATOES (Potatoes, Instnt)
3	Q 57.01	DUCHESS POTATOES (Potatoes, Instnt)
3	Q 57.02	GRILLED POTATO CAKES (Potatoes, Instnt)
3	Q 57.03	MASHED POTATOES WALDORF (Potatoes, Instnt)

1	Q 59	GERMAN SAUERKRAUT
3	Q 60	CLUB SPINACH
1	Q 61	BAKED HUBBARD SQUASH
1	Q 61.01	BAKED ACORN SQUASH
2	Q 62	CREOLE SUMMER SQUASH
	Q 63	FRIED SUMMER SQUASH
	Q 64	LOUISIANA STYLE SMOTHERED SQUASH
2	Q 65	SEASONED SUCCOTASH
2	Q 65.01	SEASONED SUCCOTASH (Frz Succotash)
	Q 66	BAKED SWEET POTATOES
3	Q 67	CANDIED SWEET POTATOES
3	Q 67.01	GLAZED SWEET POTATOES
3	Q 67.02	GLAZED SWEET POTATOES (Blended Syrup)
	Q 69	MASHED SWEET POTATOES
	Q 69.01	SWEET POTATOES SOUTHERN STYLE
	Q 69.02	MARSHMALLOW SWEET POTATOES
3	Q 70	SCALLOPED SWEET POTATOES & APPLES
3	Q 73	STEWED TOMATOES
3	Q 73.01	STEWED TOMATOES W/CROUTONS
3	Q 74	TURNIPS & BACON
1	Q 75	REFRIED BEANS W/CHEESE
1	Q 75.01	REFRIED BEANS W/CHEESE (Pinto Beans, Cnd)
1	Q 75.02	REFRIED BEANS (Cnd Refried Beans)
1	Q 75.03	REFRIED BEANS
3	Q 77	PARSLEY BUTTERED POTATOES
3	Q 77.01	PAPRIKA BUTTERED POTATOES
3	Q 80.01	BROCCOLI COMBO
3	Q 80.02	BEAN COMBO
3	Q 80.03	CAULIFLOWER COMBO
3	Q 80.04	BRUSSELS SPROUTS COMBO
3	Q 80.05	GREEN BEAN COMBO
3	Q 80.06	CORN COMBO
	Q 81	RATATOUILLE (Country Style Eggplant & Zucchini)
2	Q 82	HERBED GREEN BEANS
2	Q 83	BAKED POTATOES W/CHILI TOPPING
2	Q 83.01	BAKED POTATOES W/CHILI (Chili Con Carne w/Beans) TOPPING
2	Q 83.02	BAKED POTATOES W/CHILI (Chili Con Carne w/Beans [Dehy]) TOPPING
2	Q 83.03	BAKED POTATOES W/CHEESE SCE TOPPING
2	Q 83.04	BAKED POTATOES W/BROCCOLI & CHEESE SCE TOPPING
2	Q 83.05	BAKED POTATOES W/HAMBURGER STROGANOFF TOPPING
1	Q 84	CARROTS AMANDINE
2	Q 85	TANGY SPINACH
2	Q 86	TEMPURA FRIED BROCCOLI
2	Q 86.01	TEMPURA FRIED CARROTS
2	Q 86.02	TEMPURA FRIED CAULIFLOWER
2	Q 86.03	TEMPURA FRIED EGGPLANT
2	Q 86.04	TOEMPURA FRIED SWEET PEPPERS
2	Q 86.05	TEMPURA FRIED ZUCCHINI SQUASH
2	Q 87	HERBED BROCCOLI

CHAPTER 8.0

COMPARISON OF CAN, UMASS and LSU NUTRITIONAL ANALYSIS SYSTEMS

Contributing Authors:
Carol Baker
Dr. Kenneth Samonds
Doris Sherman

Testing and validation of the recipe analysis system will include the following comparisons of the new system with the previously used system: 1) the USARIEM nutrient database (NDB) with the NDB previously used at the University of Massachusetts, 2) nomenclature of the databases, 3) the recipe coding and calculation procedures, and 4) the results of dietary intake computations.

Recipe analyses performed on the Military Nutrition Division CAN system will be compared to analyses performed on the University of Massachusetts (UMASS) and the Louisiana State University (LSU) Extended Table of Nutrient Values (ETNV) nutrient databases and recipe analysis systems. Specifically, six representative recipes coded for the Fort Jackson garrison dining hall study and previously analyzed by the UMASS system will be reanalyzed using the new CAN system. The same recipes with detailed information collected about the ingredients used in the kitchen have been sent to LSU for coding and analysis using ETNV. Comparison of recipe analysis between the USARIEM and LSU systems is being performed because of nutrition intervention projects to be conducted by LSU at Army installations. Comparison of the two systems will allow contrast of these studies with dietary intake surveys previously conducted by USARIEM.

An additional ten recipes from the Armed Forces Recipe File, which have been chemically analyzed for nutrient content, will be computer analyzed using the CAN system, the UMASS system and LSU's ETNV. Discrepancies between the systems would reflect differences in the 1) coding of the food items due to availability of food choices, completeness of description of the food choices, and coder judgement 2) differences in the food composition data and 3) mistakes in calculation procedures. Results of chemical analysis can assist in the investigation of a questionable computer analysis, but direct validation of computer analyses cannct be made by comparison to chemical analyses of single samples. A single chemical analysis does not account for the normal range of variation of the constituents of food due to real differences in composition. Food tables are constructed to report average food values, i.e. representative figures of a year round, national food supply.

Finally, in order to ascertain the effects of the new database and analysis system on the results of a dietary assessment, the entire Fort Jackson study will be reanalyzed. All recipes in the Fort Jackson recipe file will be recoded and reanalyzed by the CAN system. Nutrient data for single item foods from the new database will replace data in the study table that was obtained from the UMASS NDB. Dietary intake for the entire seven day study period will be reanalyzed using the newly created study table and the CAN system programs. By accounting for the inevitable bias of the new system, future USARIEM studies can be compared to past studies.

Testing of the CAN system is currently underway. With the dedication of manpower resources, it is anticipated that testing will be completed by October 1990.

SUMMARY

The Computerized Analysis of Nutrients (CAN) system that the Military Nutrition Division uses to enter food consumption data into computer files, to analyze recipe information, and to combine food consumption and nutritional analysis information for its field and garrison dietary assessment studies is documented in this technical report. User's and programmer's manuals are provided to assist the computer clerks and scientists who are using the system. The capabilities of this system and the documentation of the system will change as the system is updated and more efficient methods of achieving our mission are realized.

APPENDICES

APPENDIX A

Nutrient List

Number	Name	Unit	Abbreviation
1	Water	gm	H2O
2	Kilocalories	kcal	CAL
3	Protein	gm	PROT
4	Fat, total	gm	FAT
5	Carbohydrate	gm	CHO
6	Sucrose	gm	SUC
7	Other sugars	gm	SUG
8	Complex carbohydrate	gm	COM
9	Crude fiber	gm	CRF
10	Dietary fiber, total	gm	DFIB
11	Insol. fiber	gm	DFI
12	Sol. fiber	gm	DFS
13	Ash	gm	ASH
14	Ascorbic acid	mg	C
15	Thiamin	mg	THI
16	Riboflavin	mg	RIB
17	Niacin	mg	NIA
18	Vitamin B6	mg	B6
19	Folacin	mcg	FOL
20	Vitamin B12	mcg	B12
21	Biotin	mcg	BIO
22	Pantothenic acid	mg	PAN
23	Vitamin A	RE	ARE
24	Vitamin A	IU	AIU
25	Beta carotene	RE	CARRE
26	Beta carotene	mg	CAR
27	Vitamin D	mcg	D
28	Vitamin E, total	mg	ETOT
29	Tocopherol, total	mg	TTOC
30	Tocopherol, alpha	mcg	TOC
31	Vitamin K	mcg	VITK
32	Calcium	mg	CA
33	Phosphorus	mg	P
34	Magnesium	mg	MG
35	Iron, total	mg	FE
36	Iron, heme	mg	FEH
37	Iron, non-heme	mg	FEN
38	Zinc	mg	ZN
39	Iodine	mcg	I
40	Copper	mg	CU
41	Manganese	mcg	MN
42	Fluoride	mg	FL
43	Chromium	mcg	CR
44	Selenium	mcg	SE
45	Molybdenum	mg	MO
46	Potassium	mg	K
47	Sodium	mg	NA
48	Chloride	mg	CL
49	Sulfur	mg	S
50	Aluminum	mg	AL
51	Barium	mg	BA
52	Boron	mg	BO
53	Strontium	mg	SR

Number	Name	Unit	Abbreviation
54	Silicon	mg	SI
55	Vanadium	mg	VA
56	Cobalt	mg	CO
57	Nickel	mg	NI
58	Arsenic	mg	ARS
59	Tin	mg	TIN
60	Saturated fat	gm	SAT
61	Butyric	gm	S4
62	Caproic	gm	S6
63	Caprylic	gm	S8
64	Capric	gm	S10
65	Lauric	gm	S12
66	Myristic	gm	S14
67	Palmitic	gm	S16
68	Stearic	gm	S18
69	Monounsaturated fat	gm	MON
70	Palmitoleic	gm	M16
71	Oleic	gm	M18
72	Gadoleic	gm	M20
73	Erucic	gm	M22
74	Polyunsaturated fat	gm	POL
75	Linoleic	gm	P182
76	Linolenic	gm	P183
77	Octadecatetraenoic	gm	P184
78	Arachidonic	gm	P204
79	Eicosapentaenoic	gm	P205
80	Clupadonic	gm	P225
81	Docosahexaenoic	gm	P226
82	Cholesterol	mg	CHOL
83	Phytosterols	mg	PHY
84	Histidine	gm	HIS
85	Isoleucine	gm	ILE
86	Leucine	gm	LEU
87	Lysine	gm	LYS
88	Methionine	gm	MET
89	Phenylalanine	gm	PHE
90	Threonine	gm	THR
91	Tryptophan	gm	TRY
92	Valine	gm	VAL
93	Alanine	gm	ALA
94	Arginine	gm	ARG
95	Aspartic	gm	ASP
96	Cystine	gm	CYS
97	Glutamic	gm	GLU
98	Glycine	gm	GLY
99	Proline	gm	PRO
100	Serine	gm	SER
101	Tryosine	gm	TYR
102	Met+Cys	gm	MC
103	Tyr+Phe	gm	TP
104	Caffeine	gm	CAF
105	Ethanol	gm	ETH
106	Carnitine	mcg	CARN
107	Inositol	mg	INO
108	Phytate	mg	PHT
109	Choline	mg	CHL

APPENDIX B

The Fields of the Data File

Listed below are all the fields that the FOODDE program accommodates. Any given study may be comprised of all or a subset of these fields. However, the order that is described below is the order that they will be internally stored in the output data file.

A "*" indicates that there is information concerning this field that can be accessed at the time of entry.

1.	STUDY	-5 alphanumeric characters, name of the study ie. T8901.
2.	DATE	-6 characters, date YYMMDD of consumption.
3.	DAY	-2 characters, indicates the day of week computer generated from the date field.
4.	STUDY DAY	-Integer, indicates numerically the study day.
5.	CYCLE	-Integer, indicates numerically cycle day.
*6.	GROUP	-Integer, the numeric group number of subject.
7.	SUBJECT	-Integer, the numeric subject number.
8.	SEX	-1 character, indicating sex of the subject (M or F).
*9.	MEAL	-Integer, number corresponding to a meal of the day, numbering system explained below.
10.	TIME	-4 characters, military time that meal was eaten.
*11.	SOURCE	-Integer,number corresponding to location where food was obtained/eaten, numbering system explained below.
12.	FOODNUM	-Integer, number that indicates a food item in the COD file.
13.	AMOUNT TAKEN	-Real, amount of the food item taken by the subject.
14.	AMOUNT CONSUMED	-Real, amount of the food item actually consumed.
15.	AMOUNT RETURNED	-Real, amount of the food item returned (not consumed).
*16.	REASON	-Integer, number representing a reason that subject returned part of the food item, numbering system explained below.
*17.	RATING	-Integer, number used to rate hedonically the food item consumed, rating system explained in detail below.
18.	SALT ADDED	-Real, amount of salt added to food item
*19.	DATA COLLECTOR	-Integer, number representing the specific data.

collector who collected the raw data.

A food data record is stored in the data file in the following format; a C means a character field, an I means an integer field, a R means a real number field. The number after the letter indicates the length in digits. With a real number, the number after the decimal is the number of digits stored after the decimal in the file:

C5,C6,C2,I10,I10,I10,I10,C1,I10,C4,I10,I10,R8.2,R8.2,R8.2,I10,I10,R8.2,I10

Note that the AMT fields are independent of one another: they do not interact to calculate values for the other, so they must be entered separately. They also do not represent a standard unit, say grams for instance. They represent ONE unit of the food item. The unit is defined in the COD file.

As indicated by the *, some fields have more information concerning them, which can be accessed as on-line help. There are 30 possible correspondences for each of these fields, but most of them use a much smaller number. The translation of the number entered to the real meaning of this value are described below for the "*" fields.

MEAL:

1	BREAKFAST
2	MID DAY
3	SUPPER
4	BRUNCH
5	SNACK
6	BEVERAGE
7	OTHER
8-30	Undefined

SOURCE:

1	DINING HALL-SHORT ORDER LINE
2	DINING HALL-MAIN LINE
3	RESTAURANT
4	CLUBS/BARS
5	SNACK BAR
6	FAST FOODS
7	HOME
8	BARRACKS
9	VENDORS
10	VENDING MACHINE
11	OTHER
12-30	Undefined

RATING:

1	DISLIKE EXTREMELY
2	DISLIKE VERY MUCH
3	DISLIKE MODERATELY
4	DISLIKE SLIGHTLY
5	AMBIVALENT
6	LIKE SLIGHTLY
7	LIKE MODERATELY
8	LIKE VERY MUCH
9	LIKE EXTREMELY
10-30	Undefined

REASON:

1	NOT HUNGRY
2	NO TIME
3	DIDNT LIKE
4	TOO MUCH
5	FULL
6	COLD
7	SPICY
8	BUGS
9	ILL
10	MUST TAKE
11	RAW
12	OVERCOOKED
13	BURNT
14	SAVE IT
15	ON DIET
16	SERVED A LOT
17	GAVE AWAY
18	ROTTEN
19	NOT THIRSTY
20	SWEET
21	UNKNOWN
22-30	Undefined

The GROUP and DATA COLLECTOR translations are defined by the user and stored in the FOOD.DEF file. Consequently, they can be study specific values. See section 4.2.3.5 for instructions on how this is accomplished.

APPENDIX C

The COD File

The COD (short for CODE) file is an external file that the FOODDE program reads in to obtain the list of all the valid Food Items used in the study. The COD file is created prior to the data entry phase, and it is usually updated due to the consumption of items that were not initially included in the file. Below is the description of a COD file and all its fields:

COD CODE	3 digits. Assigned number that generally corresponds to a food item's line position in the COD file. For example, the food item with a Cod code equal to 27 would be on the 27th line of the COD file. The COD number is how the data enterer chooses a food item, and it makes for quicker data entry.
FOOD CODE	6 digits. The code by which the nutrient database refers to the item. This code will be used to reference a food item's nutrient data.
FOOD NAME	45 digits. The descriptive name of the food item.
UNITS	10 digits. The unit by which the item is served (ounces, cups, gallons, etc.).
GRAM WEIGHT	Real number. The gram weight of a unit of the food.
GROUP CODE	8 digits. Describes the food group or category to which the food item belongs.
ENTRY DATE	DD-MM-YY. The date that the food item was entered into the COD file.

APPENDIX D

The Template File

The TEMPLATE file governs how data will be entered. Prior to a study, the investigators should decide what fields need to be activated, their order, whether or not to make it a default field, and whether or not to specify a range for it. The TEMPLATE file is an unformatted file that contains one record. This one record contains all the descriptive information about the fields. For every field (in the order specified in Appendix B), the TEMPLATE file will contain:

PROMPT	-	The prompt that appears for the field when entering data for it.
PLACE	-	The position, 1 through 19, of the screen template that the field occupies. A zero indicates an inactive field.
LOW_VAL	-	The low range value for the field, meaningful only ifthe RANGE field is set to TRUE.
HI_VAL	-	The high range value for the field, meaningful only if the RANGE field is set to TRUE.
RANGE	-	Set to TRUE if user wishes to specify a range for the field, or FALSE for no range specification.
ACTIVE	-	Set to TRUE if field is active for this template, or FALSE if user chose not to use the field.
DEFAULT	-	Set to TRUE if the user wants the field to be a default field, or FALSE if it is not a default.

The STUDY field is automatically made a default field by the program if it is activated by the user.

The DAY field is dependent upon the DATE field. The user will be prompted to activate it if the DATE field is activated, and its default status is determined by the DATE field default value.

APPENDIX E

Diet History Data File

The Diet History data file is a fortran keyed indexed file that uses the first field, the line number, as the key. The file can be typed to the terminal screen, but can only be edited through the use of the Diet History Analyzer program. The fields of the data file are listed below:

LINE NUMBER	Sequential character number that is the actual line number of the record in the data file.
SUBJECT	The subject number assigned to the subject.
AGE	The age of the subject.
SEX	The sex of the subject.
DATE	The date that the food item was consumed by the subject.
TIME	The time of day (military format) that the food was consumed.
MEAL	Number corresponding to meal type; For example, 1=breakfast.
FOOD CODE	The nutrient database food code of the food item consumed.
UNITS LABEL	The unit label, up to fifty characters, for the food item.
AMOUNT	The amount, in terms of units, that the subject consumed.
WEIGHT	The gram weight for the amount of the food consumed.

The FORTRAN format used to store a diet history data record is as follows:

```
A3,1X,A4,1X,A2,1X,A1,1X,A6,1X,A4,1X,A3,1X,A10,1X,A50,1X,F9.2,1X,F10.1
```

APPENDIX F

Files Created By Universal Data Entry Program

After running every option in the Main Menu, a total of six files will exist in the directory pertaining to the data file. All will have the same file name, but will have different dot extensions. These files are as follows:

(1) Filename.DAT (essential) - the file that contains all the data that you entered. It is created by option 1 on the Main Menu and is updated by options 1, 2, 3, 5, and 6. Should not be deleted.

(2) Filename.FMT (essential) - this file contains all the information about the variable formats. It is created by option 1 and updated by options 1, 5, and 6. Should not be deleted.

(3) Filename.TXT (not essential) - this file contains a list of the variables in the data file, their FORTRAN formats, and the columns that they are in. It is created by options 1, 5, and 6. This file is not used in running the program and can be deleted at any time, but a printout should be made for future reference.

(4) Filename.REP (not essential) - this file contains a formatted printout of the data file. It is created by option 4. This file can be deleted at any time.

(5) Filename.DFT (not essential) - this file contains the customized default headings to be used when a report is produced. This file is not necessary when running the program, but the customized headings are lost and will need to be recreated. It is created by option 4 and erased by option 6.

(6) Filename.TRE (semi-essential) - this file contains information about which records in the file have been updated with new variables. It is created by option 5 and updated by option 5. Once all new values have been entered, the file can be deleted. If deleted too soon, information on which records have been updated will be lost.

APPENDIX G

Program Listing and Technical Information

G-1 General Information

The following sections discuss aspects of the system which require more in-depth explanations. When referring to the attached programs, line numbers will be given. These numbers run the length of the page in the left-most columns, starting with "0001." These number should not be confused with label numbers, which occur in the columns between the line numbers and the body of the program.

In the actual file containing the program, the first 5 spaces of a file are reserved for label numbers and comment indicators. The sixth space is reserved for a character indicating the current line is a continuation of the previous one. Spaces 7 through 72 are used for program commands while all spaces beyond 72 are ignored. The labels are used as reference points, may appear in any increment or order, and are the target of commands such as GOTO. By convention there is an increment of 10, to allow for unexpected reference points. Also by convention, the label numbers should be increasing as you move down the program. Before an explanation of key sections, there is some background information you need to know:

G-1.1 Variables

1. Variables should be named briefly yet appropriately. Standard FORTRAN limits variable names to 6 characters, but the VAX allows for much larger ones. If you are positive that the program will never be run on a PC, then the names may be longer than 6 characters. Also try to give variables names that have a meaning relating to the context in which they are used (i.e. do not use 'X' as a variable name.) Each variable used should be declared at the top of the program in the variables section. This is very important because FORTRAN does not require that variables be declared in order to use them. If a variable is not declared then the initial letter of the name determines the variable type ("A"-"H" and "O"-"Z" indicate real variables, "I"-"N" indicates integer variables.) Thus, if you do not declare a variable, the program will still execute but the variable will be defined by its name. This could lead to improper handling of the data or an error when running the program.

2. Character variables may contain any alphanumeric value. They cannot have math operations performed on them. Also, you must specify the maximum possible length that any value of the variable could have. This is done by

placing an asterisk (*) after the variable, followed by the number of characters signifying the length of the variable. For example: "FodFile*20".

3. Integer variables may contain only whole numbers.

4. Real variables may contain either whole numbers or decimal values. If a whole number is entered, it is stored as a decimal value (i.e. 4 becomes 4.0.)

5. Logical variables may contain True/False values. You actually assign these words to the variable to give it a value.

G-1.2 Formats

1. Format statements describe the organization of a file or output device (i.e. screen) to the program. Whenever you are reading or writing information you need to specify a format to the program. Each format must have a unique label number and may occur anywhere in the program. Some programmers place format statements at the end of the program in one body while others place each statement near the read or write statement using that format.
2. The format of a format statement is as follows:

<label #> FORMAT(<format description>)

The label number occurs in columns 1 through 5, is unique in the subroutine or main program, consists solely of numeric digits, and may have preceding zeros (i.e. "00050" is equivalent to "50"). The word "FORMAT" must begin in column 7 or later. The format description consists of identifiers which describe the type and length of each variable to be processed (see below for more information on these identifiers.) If the format description is so long that it extends beyond column 72, continue it on the next line. To indicate a continuation, place an asterisk in column 6 and then finish the rest of the description, beginning at column 7. A single format statement may continue over several lines.

G-1.3 Variable Identifiers

1. The variable identifiers specify the type and length of each variable to be processed. Each identifier must be separated by a comma, and the entire list must be enclosed by parentheses. Tabs can be specified by the letter "T" followed by the number of columns from the left.

2. Character variables are indicated by the letter "A" followed by the length of the string (i.e. "A5").

3. Integer variables are indicated by the letter "I" followed by the number of digits (i.e. "I5").

4. Real variables are indicated by the letter "F" followed by the total number of digits, a decimal, and the number of digits after the decimal. For example, a real number eight digits in length with two of the eight digits after the decimal would be coded as "F8.2".

5. Spaces are indicated by the letter "X" <u>preceded</u> by the number of digits (i.e. "5X".)

6. An example of a format statement is as follows:

   01000 FORMAT(T2,A6,I5,A6,3X,F6.2,T33,I2,F6.2)

   This statement would process data in the following format (note that each variable is denoted by its associated identifier and that blanks, whether from tabs or spaces, are coded by "X"):

```
         1         2         3         4
1234567890123456789012345678901234567890
XAAAAAAIIIIIAAAAAAXXXFFF.FFXXXXXIIFFF.FF
```

G-1.4 Input/Output

1. Input and output statements, READ and WRITE, respectively, have similar arrangements, requiring a unit device, a format label, and optional error code all enclosed in parentheses and followed by the variables to be processed.

2. The unit device is a number indicating the unit specified by an open statement for the appropriate file. The values of 5 and 6 are reserved for the keyboard and screen, respectively. Any other two-digit integer value is acceptable.

3. The format label has been covered in the above section.

4. The error code is simply a label to which the program should jump in the case of an error occurring.

5. There are other options available for these statements but they are beyond the scope of this document.

G-1.5 Other Considerations

1. The premise of the above sections is to give you a basic understanding of FORTRAN in order to make simple modifications. If major alterations are needed, see a programmer.

G-2 Technical Aspects of the COD File

The COD file is an ASCII file, the contents and purpose of which are described in section 5.1.3.2.1. If corrections or additions need to be made to the COD file, there are two alternatives. First, you may simply edit the file with the system editor and manually add information. Secondly, you may use an interactive program, named CODMAKER, which automates the maintenance of the COD file and other associated tasks. The program will prompt you for the needed information.

Regardless of which method you use, there are a few concepts you should be aware of when manipulating the COD file. Probably the most important feature is the effect of the COD file's organization upon entry programs. Each food item in the COD file exists on one physical line of the file. An entry program accesses the foods by referencing the physical line the item is supposed to exist on. For example, if an item is on the 25th line of a file, it would be accessed by the code number "25". The second feature you should be aware of is that there are specific columns that the variables must exist in. In the past the column positions have varied from study to study. If you use the CODMAKER program, default columns are chosen for you, but if you do not use the program you should attempt to follow its format for future studies. Sometimes it may be more convenient to use a COD file from past studies and stay with its format. This is the case for the T8901 Alaska study. As you can see on the following page, each variable lines up in specific columns. Note that, for character variables, such as food name, the items are left justified, while for numeric variables, the items are either right justified or aligned by the decimal point. Also be aware that a character variable is one in which characters <u>may</u> appear.

```
        Database
  COD    Code                                                        Unit of         Gram        Group
   #       #                  Food Name                              Measure         Weight      Code        Date

                                                                                                         1         1
           1         2         3         4         5         6         7         8         9         0         1
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
 100  RL4406 CEREAL BAR-BRAN FLAKE JAN 88                           ONE              72.30    2DB00000  29-JUN-89
 101  RL4407 CEREAL BAR-CORN FLAKE JAN 88                           ONE              69.30    2DB00000  29-JUN-89
 102  RL4403 CEREAL BAR-GRAPENUT (GRANULES) JAN 88                  ONE              82.00    2DB00000  29-JUN-89
 103  RL4402 CEREAL BAR-LIFE (OAT) JAN 88                           ONE              66.40    2DB00000  29-JUN-89
 104  RL4408 CEREAL BAR-WHEATIES (WHEAT) JAN 88                     ONE              67.50    2DB00000  29-JUN-89
                                                     .
                                                     .
                                                     .
 145  027010 WATER ADDED TO RATIONS                                 1 FL OZ          30.00    20000000  29-JUN-89
 147  999999 PARMESAN CHEESE                                        TBSP              5.00    20000000   9-AUG-89
 148  999998 GATORADE                                               1 FL OZ          30.00    2ICA0000   9-AUG-89
 149  999997 APPLE CIDER                                            1 FL OZ          30.00    2ICB0000   9-AUG-89
 150  999996 HOT DOG, FRANK, BEEF                                   1 EA             45.00    3BD00000   9-AUG-89
```

G-3 The Change Program

G-3.1 General Information

The CHANGE program used as an example is that developed for the T8901 Alaska study. It is short and fairly straight forward. The program can be broken down into four main sections: variable declaration, data initialization, data processing, and format specification. Depending upon the complexity and scope of changes that have to be make for the study you are dealing with, you may have to make few changes in this program. If this is the case, you may only need to change the directories and names of the files in the open statements. It is conceivable, however, that major revisions will have to be made. In this case, contact the resident programmer or the Information Management Branch.

G-3.2 Variable Declaration Section

The variable declaration section, in lines 3 through 11 lists all of the variables used by the program, defining each by its type and, if appropriate, length. The variables FdNam, FdCod, and TGram are arrays of 200 elements (i.e. 200 items can be stored in each). For character variables, the length is specified after the asterisk. Thus, FdCod can hold 200 items, each six characters long, while Group can hold only one character.

G-3.3 Data Initialization Section

The data initialization section, lines 13 through 56, accesses and prepares the data for manipulation. It has several different functions:

1. Lines 13 through 19 assign appropriate values in element "200" for an item that was not eaten but needs to be considered in analysis. Normally the code would be a value referencing a line of the COD file, but for this study a value of "0" was assigned. Since there is no line "0" of a file, the last element of the array was assigned, and "0" is replaced by "200". This is a one-time problem and not applicable to other studies, but it is a good example of the adjustments that need to be made in order to analyze the data.

2. Lines 21 through 33 fill the FdNam, FdCod, and TGram arrays, getting the information from the COD file. Notice on line 26 the full directory path is included in the file name. This is done to ensure that the correct files are accessed and that new files are created in the proper directory. How this section works is fairly straightforward: the three-digit code is read into T3Code, the computer goes back to the same line (via the command "Backspace"), and reads in the six-digit food code, the food name, and the

unit gram weight, placing each value in the element of the array specified by the three-digit code. For example, if the three-digit code read in was 123, then the values read in on line 31 would be placed in element 123 of the arrays. Notice that the command on line 33 sends the computer back to line 29 (the statement "Goto 10" refers to the label 10, not the line number.) This causes an unconditional loop (i.e. one that cannot be exited), except for the statement "End=15" on line 29. This tells the program to goto label 15 (line 39) when it encounters the end of the COD file. By the time the programs reaches label 15, the entire COD file will have been read in the various arrays, each one indexed by the three-digit code.

3. 5.4.3.1 Lines 35 through 56 open the master data files FOODS.TOT and RATING.DAT along with the file NAMES.LIS. The only difference between FOODS.TOT and RATING.DAT is that the latter does not contain the amounts of foods eaten. FOODS.TOT will be used in nutrient analysis while RATING.DAT will be given to the Food Engineering Directorate for their own analysis. The file NAMES.LIS contains the names of the data files that were created by the data entry program. These files are maintained for correction purposes; it is easier to work with small specific files than an unwieldy master file. These three main files are discussed more in-depth in section 5.1.5.4.3.1.

G-3.4 Data Processing Section

The data processing section, lines 58 through 95, updates and reformats the information into a organization better suited for analysis. It has the following sections.

1. Lines 58 through 70 read a data file name from NAMES.LIS, open the data file, and extract the date and group values from the file name. Note that, on line 66, the file is given a status of old. This indicates that the file already exists. If this is not the case, the program will crash. Also note that, although all information is usually stored within the data files themselves, in this case the date and group information was captured only in the name. As the statistics packages usually require information of this type to be contained in each record, these values will be placed within the master data file. Lines 69 and 70 extract the values, the date from characters 7 and 8 of the file name and the group from character 3.

2. Lines 72 through 78 read in a record from the raw data file and make sure to change the value of code to 200 if its original value was 0. Note in the read statement on line 77, that when the program gets to the end of the raw data file, it is to go to label 20 (line 64) to get a new file name.

3. Lines 80 through 90 write the information to the two master files, FOODS.TOT and RATING.DAT and then send the program to label 30 (line

77) to get another record from the file. Note that the variables have been rearranged and that the information in the corresponding elements of the FdNam, FdCod, and TGram arrays are accessed.

4. Lines 93 through 95 close all of the file in preparation for the program to finish.

G-3.5 Format Specification Section

This section, lines 97 to 104, defines the input and output formats for the program. See Appendix G, Sections 1.2 through 1.4 for information on how to determine these specifications.

Note that line 105 has the statement "End" on it. This is a required key word indicating that the program is finished.

```
     Program Change

C********************  Variables  ********************

     Character     Fodfil*50,FdNam(200)*35,FdCod(200)*6,
     *                    TDate*2,Group*1

     Integer       RNum,SubNum,Code,Ans,Rating,RsnNE,
     *                    T3Code

     Real          AmtCons,FdGram,TGram(200)

C
C    Initialize the 'Did not Eat' variable.  Whenever the program encounters
C    this code, it writes these values to the files.
C
            FdCod(200) = 'DIDNTE'
            FdNam(200) = 'Did not Eat'
            TGram(200) = 0.00

C
C    Access the COD file and read in the information to arrays.  The index
C    of the arrays is the three-digit code (i.e. Code 95 will have its
C    information stored in element 95 of the arrays.)
C
            Open(Unit=1,File='[NUTRITION.T8901.DATA]Alaska.COD',
     *           Status='Old',
     *       Access='Sequential',Organization='Sequential')
10   Read(1,1000,End=15)T3Code
            Backspace(1)
            Read(1,1010)FdCod(T3Code),FdNam(T3Code),TGram(T3Code)

            Goto 10

C
C    Open the master files to which all records are written.  FOODS.TOT
C    is complete while RATING.DAT lacks consumption values.
C
15   Open(Unit=4,File='[NUTRITION.T8901.DATA]Foods.tot',
     *             Status='New',RecordType='Fixed',
     *       Access='Sequential',RecL=132,Form='Formatted',
     *       Organization='Sequential')

        Open(Unit=7,File='[NUTRITION.T8901.DATA]RATING.DAT',
     *             Status='New',RecordType='Fixed',
     *       Access='Sequential',RecL=60,Form='Formatted',
     *       Organization='Sequential')


C
C    Open the file containing the list of data-file names which need to be
C    accessed and processed.
C
            Open(Unit=2,File='[NUTRITION.T8901.DATA]Names.lis',
     *             Status='Old',
     *             Access='Sequential',Organization='Sequential')

C
C    Read in a file name, open the file, and extract from the file name
C    the date and group values. If, when reading the file name, the end
C    of the file is encountered, then all of the files have been processed
C    and the program can end.
```

```
C
20    Read(2,1020,End=40)FodFil

      Open(Unit=3,File=Fodfil,Status='Old',
      *             Access='Sequential',
      *             Organization='Sequential')
            Tdate = FodFil(7:8)
            Group = FodFil(3:3)

C
C     Read a record from the file. If the end of the file is encountered,
C     then goto label 20 and read in a new file name.  If the CODE = 0 then
C     re-code it to 200 (the value for 'Did not Eat'.)
C
30    Read(3,1030,End=20)SubNum,Code,AmtCons,Rating,RsnNE
            If (Code .EQ. 0) Code = 200

C
C     Write the information to FOODS.TOT and RATING.DAT, using the values
C     read in from the COD file for the food name, food code, and gram
C     weight.  Then goto label 30.
            Write(4,1040)Group,TDate,Subnum,Rating,RsnNE,Code,
      *             FdNam(Code),FdCod(Code),TGram(Code),AmtCons
            Write(7,1040)Group,TDate,Subnum,Rating,RsnNE,Code,
      *             FdNam(Code)


      GO TO 30


40    Close(Unit=2,Status='Keep')
      Close(Unit=3,Status='Keep')
      Close(Unit=4,Status='Keep')

C*********************** FORMATS ******************************

1000  Format(T2,I3)
1010  Format(T7,A6,T23,A35,T58,F6.2)
1020  Format(A)
1030  Format(T5,I3,T10,I3,T20,F6.2,T32,I1,T50,I2)
1040  Format(T2,A1,1X,A2,1X,I3,1X,I1,1X,I2,1X,I3,1X,
      *             A35,1X,A6,1X,F6.2,1X,F6.2)
         End
```

DISTRIBUTION LIST

	NO. OF COPIES
Defense Technical Information Center ATTN: DTIC-DDA Alexandria, VA 22304-6145	12
Commander U.S. Army Medical Research and Development Command	
SGRD-RMS	1
SGRD-PLC Fort Detrick Frederick, MD 21701-5012	1
Commandant Academy of Health Sciences, U.S. Army	
ATTN: AHS-CDM	1
ATTN: HSHA-CDM	1
ATTN: HSHA-CDS Fort Sam Houston, TX 78234	1
Dir of Biol & Med Sciences Division Office of Naval Research 800 N. Quincy Street Arlington, VA 22217	1
CO, Naval Medical R&D Command National Naval Medical Center Bethesda, MD 20014	1
HQ AFMSC/SGPA Brooks AFB, TX 78235	1
Under Secretary of Defense Research and Engineering ATTN: OUSDRE(RAT)E&LS Washington, DC 20310	1
Dean School of Medicine Uniformed Services University of Health Sciences 4301 Jones Bridge Road Bethesda, MD 20014	1

DISTRIBUTION LIST (continued)

	NO. OF COPIES
Commander U.S. Army War College Carlisle Barracks, PA 17013	1
Commander U.S. Army Soldier Support Center Ft. Benjamin Harrison, IN 46216	1
Assistant Secretary of Defense (Health Affairs) ATTN: ASD(HA) PA&QA Washington, DC 20310	1
Assistant Secretary of Defense (Aquisition & Logistics) ATTN: OASD(A&L)SD Washington, DC 20310	1
Commander U.S. Army Troop Support Command ATTN: AMSTR-E 4300 Goodfellow Boulevard St. Louis, MO 63120-1798	1
Commander U.S. Army Test and Evaluation Command ATTN: AMSTE-EV-S Aberdeen Proving Ground, MD 21005-5055	1
Commander U.S. Army Operational Test Evaluation Agency ATTN: CSTE-ZX 5600 Columbia Pike Falls Church, VA 22041	1
Commander U.S. Army Training and Doctrine Command ATTN: ATCD-S Fort Monroe, VA 23651	1
Commander U.S. Army TRADOC Combined Arms Test Activity ATTN: ATCT-PO Ft. Hood, TX 76544	1

DISTRIBUTION LIST (continued)

	NO. OF COPIES
Commander	
U.S. Army Natick Research, Development and	
Engineering Center	
ATTN: STRNC-W	1
ATTN: STRNC-Y	1
ATTN: STRNC-T	1
ATTN: STRNC-E	1
ATTN: STRNC-TAA	1
ATTN: STRNC-YBH (Ms. Prell)	1
Natick, MA 01760-5000	
HQ U.S. Marine Corps	1
Code LFS-4	
Washington, DC 20380-0001	
Dept of Clinical Investigation	2
Chief, Army Medical Specialist Corp-CIS	
WRAMC	
Washington, DC 20307-5001	
MAJ Robert Stretch	2
DCIEM	
1133 Sheppard Ave. West	
P.O. Box 2000	
Downsview, Ontario, Canada M3M 3B9	
HQDA, OTSG	2
ATTN: SGPS-FP	
Suite 608	
5111 Leesburg Pike	
Falls Church, VA 22041-3258	
Commander	
John F. Kennedy Special Warfare Center and School	
ATTN: ATSU-CD-TE	1
ATTN: ATSU-CD-ML-M	1
ATTN: DOCD-M-L	1
Fort Bragg, NC 28307-5000	
Commander	1
U.S. Army Medical Research and Development Command	
ATTN: SGRD-ZB	
Fort Detrick	
Frederick, MD 21701-5012	

DISTRIBUTION LIST (continued)

	NO. OF COPIES
USDA, ARS Human Nutrition Research Center ATTN: Dr. Henry C. Lukaski P.O. Box 7166 University Station 2420 2nd Ave. North Grand Forks, ND 58202-7166	1
HQ, V Corps ACofS, G1 ATTN: AETV-GAD, Lynne Man, RD, MPH	1
Health/Fitness Nutritionist ATTN: Dr. Bernadette Feist-Fite NDH-A-ED Fort McNair, DC 20319-6000	1
Mrs. Terrie Clarke, RD U.S. Army Physical Fitness School Fort Benjamin Harrison, IN	1
Dietitian Staff and Faculty - Cadet Mess US Military Academy West Point, NY 10996	1
Commander US Army Medical Research Institute of Infectious Diseases Fort Detrick, Frederick, MD 21701-5011	1
Commander US Army Biomedical Research & Development Laboratory Fort Detrick Frederick, MD 21701-5010	1
Commander US Army Medical Materiel Development Activity Fort Detrick Frederick, MD 21701-5009	1
Commander US Medical Research Acquisition Activity Fort Detrick Frederick, MD 21701-5014	1

DISTRIBUTION LIST (continued)

	NO. OF COPIES
Commander	1
U.S. Army Materiel Command	
ATTN: AMCDE-S	
Alexandria, VA 22333	
Commander	1
USAMC Installations and Services Activity	
Rock Island, IL 61299-7190	
Commander	1
U.S. Army Combined Arms Center	
ATTN: ATZL-TIE	
Fort Leavenworth, KS 66027-5130	
HQDA OTSG	1
ATTN: DASG-DBD	
Rm 617, Bldg 5 Skyline Place	
5111 Leesburg Pike	
Falls Church, VA 22041-3258	
HQDA	1
ATTN: DASG-RDZ	
Washington, DC 20310-2300	
HQDA	1
DCSLOG	
ATTN: DALO-TST	
Washington, DC 20310-2300	
Commandant	
U.S. Army Quartermaster School	
ATTN: ATSM-CDT	1
ATTN: ATSM-SFS-FM	1
Fort Lee, VA 23807	
Commandant	
U.S. Army Troop Support Agency	
ATTN: DALO-TAF	1
ATTN: DALO-TAF-F	1
FT. Lee, VA 23801	

DISTRIBUTION LIST (continued)

	NO. OF COPIES
Commander US Army Institute of Dental Research Washington, DC 20307-5300	1
Commander US Army Medical Research Institute of Chemical Defense Aberdeen Proving Ground, MD 21010-5425	1
Commander Walter Reed Army Institute of Research Washington, DC 20307-5100	1
Commander US Army Institute of Surgical Research Fort Sam Houston, TX 78234-6200	1
Commander US Army Aeromedical Research Laboratory Fort Rucker, AL 36362-5000	1
Commander Letterman Army Institute of Research Presidio of San Francisco, CA 94129-6800	1
Commander USAMC Installations and Services Activity ATTN: AMXEN-M Rock Island, IL 61299-7190	